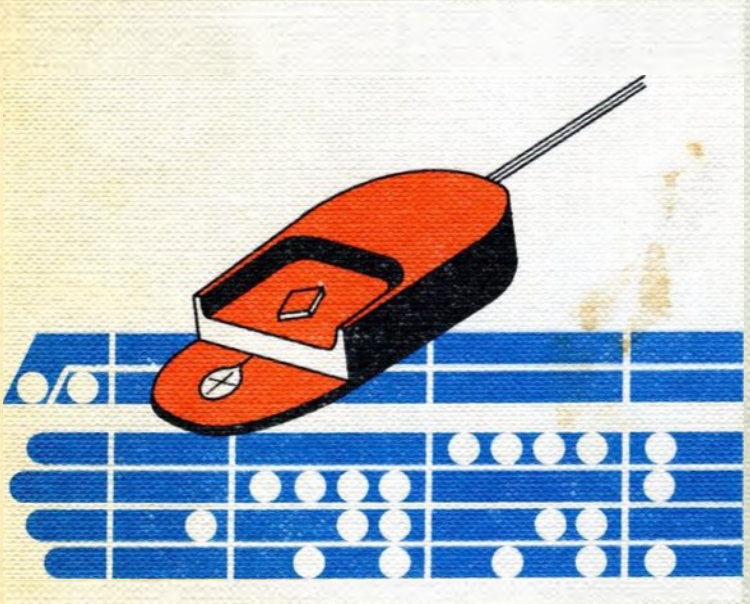
Klein
Mikrocomputer Hard- und Software-praxis

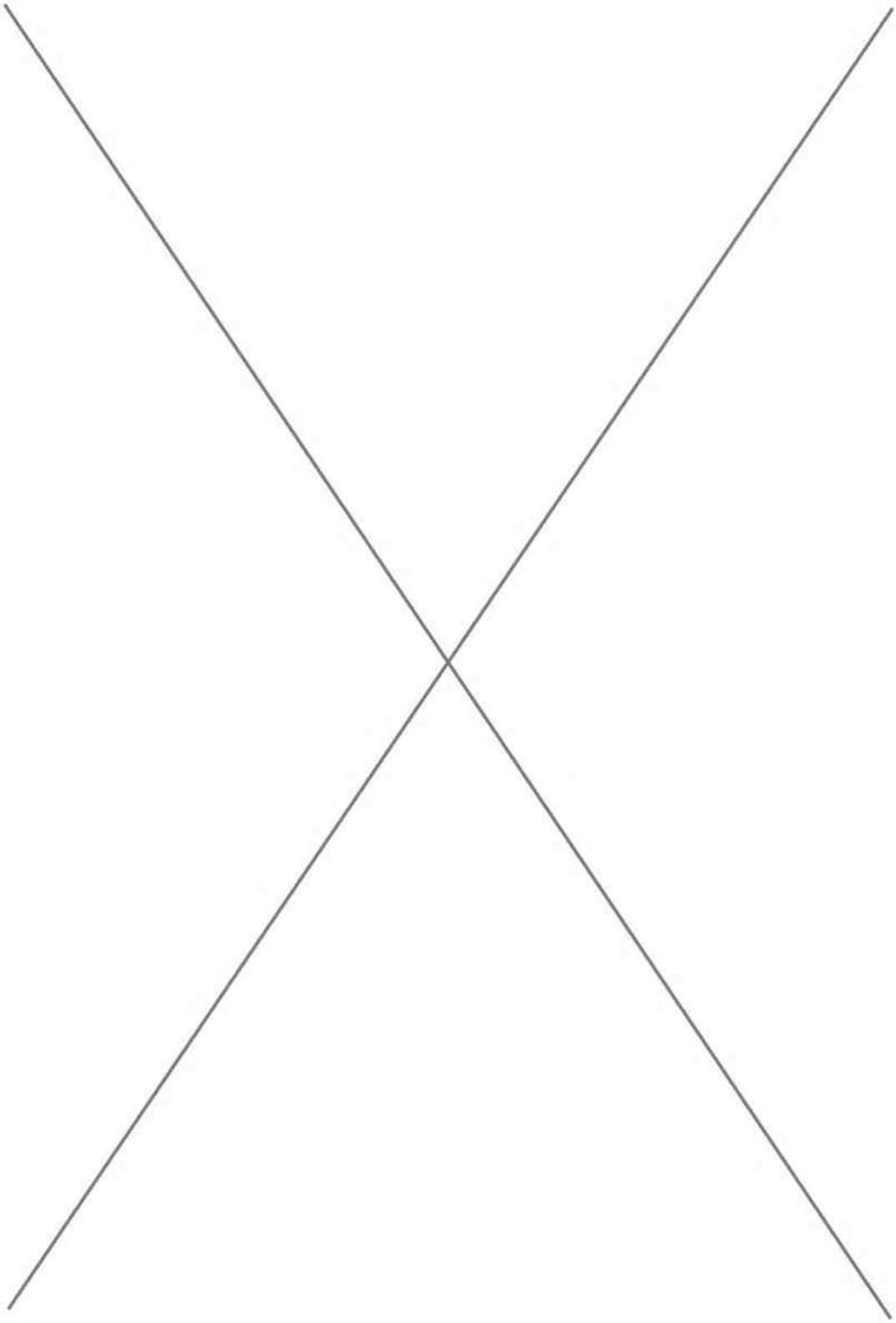
Anhand ausführlicher Beispiele und größerer Programme wird das Programmieren immer perfekter



Z80-Aufbau und Befehlssatz
Mikrocomputersystem mit dem Z80
„power on jump"
Serienports
Peripherieschaltungen
Kassettengerät als Datenspeicher
Besondere Peripheriesysteme
Monitorprogramme
Editor
Assembler
BASIC-Befehlssatz
Disassembler
Testhilfen für Software
Befehle des Tracers
Lader für das TDL
Relocating-Format
CP/M-Betriebssystem
Boolscher Formelanalysator

Franzis'

Klein, Mikrocomputer
Hard- und Softwarepraxis



Rolf-Dieter Klein

Mikrocomputer Hard- und Softwarepraxis

Anhand ausführlicher Beispiele und
größerer Programme wird das Programmieren
immer perfekter

Mit 125 Abbildungen und 6 Tabellen

Franzis'

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Klein, Rolf-Dieter:
Mikrocomputer, Hard- und Softwarepraxis: anhand ausführl. Beispiele u. größerer Programme wird d. Programmieren immer perfekter / Rolf-Dieter Klein. – München: Franzis, 1981.
ISBN 3-7723-6811-5

© 1981 Franzis-Verlag GmbH, München

Sämtliche Rechte, besonders das Übersetzungsrecht, an Text und Bildern vorbehalten.
Fotomechanische Vervielfältigungen nur mit Genehmigung des Verlages.
Jeder Nachdruck – auch auszugsweise – und jegliche Wiedergabe der Bilder sind verboten.

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany · Imprimé en Allemagne

ISBN 3-7723-6811-5

Vorwort

Schon bei der Entstehung des im gleichen Verlag erschienenen Buches des Verfassers „Mikrocomputersysteme" zeigte sich eine Stoff-Fülle, die unmöglich in einem Buch verarbeitet werden konnte. Außerdem hätte sich die Herausgabe unnötig verzögert. Der schnelle Verkauf der ersten Auflage und die Herausgabe einer zweiten Auflage rechtfertigten die Überlegungen von Verlag und Verfasser.

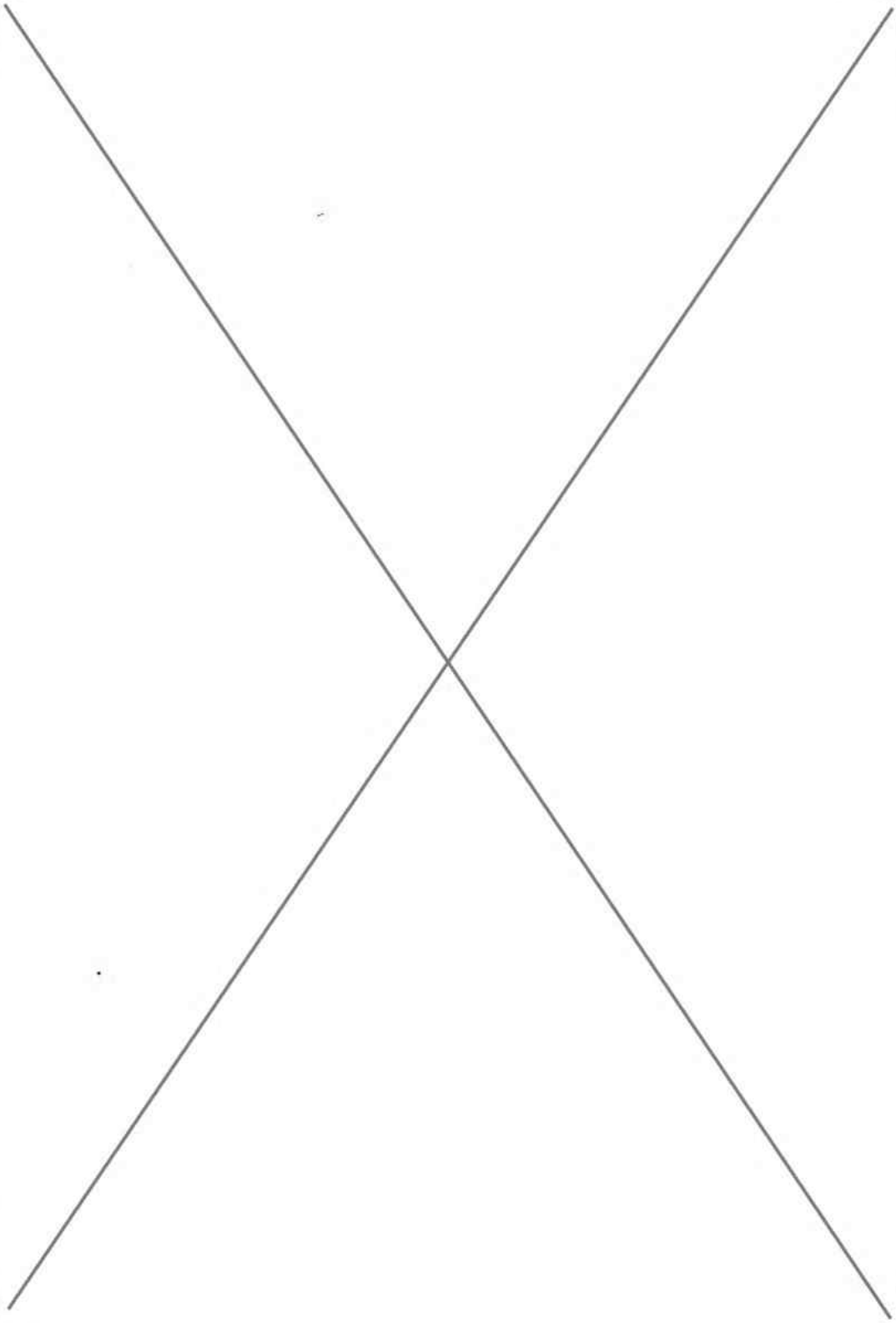
Das Vorliegen eines abgeschlossenen Stoffes für ein weiteres Buch, welches sich vor allem auf die wichtige Peripherie der Mikroprozessoren konzentriert, ermöglichte die Herausgabe dieses Werkes. Hier werden eine Reihe von Aufgaben gleichzeitig erfüllt. Die fortschreitende Technik der Mikroprozessoren erfordert weit über die Grundlagen hinaus ein Umdenken. Hierzu trägt nicht nur die Vielfalt der ständig herauskommenden hochintegrierten Bauelemente rund um den Mikroprozessor bei, sondern leider auch, das muß hier gesagt werden, die nicht immer sehr sorgfältig ausgearbeitete Dokumentation, die bei Übersetzungen selten korrigiert, sondern meist noch mit zusätzlichen Falsch-Interpretationen versehen ist. Dadurch wird es einem Anwender nicht gerade leicht gemacht, sich in die neue Technik einzuarbeiten. Das gilt natürlich ganz besonders für die Software mit den Erklärungen von Programmen für Assembler, Editoren usw.

Dieses Buch beschäftigt sich in zwei von drei Hauptabschnitten ausschließlich mit der Software, wobei aber die Hardware durchaus nicht zu kurz kommt. Die Erfahrungen, die in diesem Buch niedergelegt sind, stammen aus umfassender Praxis in einem eigenen Labor, in welchem Hardware aufgebaut und mit der erforderlichen Software versehen wurde. Alle modernen Peripheriegeräte, wie Bildschirmgerät, die verschiedensten Floppys, Cassettenrecorder und modernste Drucker wurden hierbei eingesetzt und teilweise durch eigene BUS-Systeme und Programme miteinander verknüpft. Die Originale der ausgedruckten Programme, Listings oder Aufstellungen sind in dem Buch abgedruckt und ausführlich erklärt, vor allem wenn sie ungebunden durch Systeme miteinander verknüpfte Anordnungen behandeln, beispielsweise wird an Hand eines vollständig ausgedruckten EDITOR-Programms die Arbeitsweise erklärt.

Entsprechend dem umfangreichen Inhalt dieses Buches mit Beschreibungen vom EDITOR, MONITOR, RELOCATOR, TRACER und ASSEMBLER wendet es sich an einen großen Kreis von Interessenten.

Die BASIC-Interpretation mit der Angabe des vom Verfasser entwickelten vereinfachten RDK-BASIC mit geringem Speicherbedarf zeigt dem Praktiker Möglichkeiten und gibt eine Fülle von Anregungen. So kann dieses Buch als Anregung dienen für Hobby, Studium und Beruf, um Interessenten den Zugang zu diesem faszinierenden Gebiet zu ermöglichen. Der Verfasser möchte nicht versäumen, dem Verlag für Unterstützung und Verständnis zu danken.

Rolf-Dieter Klein



Inhalt

1	**Hardware**	9
1.1	Z80 Aufbau und Befehlssatz	9
1.1.1	Z80 Hardware	9
1.1.2	Software	26
1.2	Mikrocomputersystem mit dem Z80	37
1.2.1	Speichererweiterung	37
1.2.2	„power on jump“	42
1.2.3	Parallelports	44
1.2.4	Serienports	46
1.3	Peripherieschaltungen	49
1.3.1	Display – Datensichtgeräte	49
1.3.1.1	CRT-Controler und ihre Anwendung	49
1.3.1.2	Anschluß von Zeichengeneratoren und Graphikschaltungen an den CRT-Controler	62
1.3.1.3	Restliche Baugruppen für ein CRT-Gerät	71
1.3.2	Drucker und deren Ansteuerung	76
1.3.2.1	Paralleldrucker	76
1.3.2.2	Drucker mit Serienschnittstelle	79
1.3.2.3	Drucker mit direkter Prozessorführung	81
1.3.3	Kassettengerät als Datenspeicher	84
1.3.4	A/D-Umsetzer	89
1.3.5	D/A-Umsetzer	96
1.4	Besondere Peripheriesysteme	97
1.4.1	Writehander	98
1.4.2	Digitalisierer	99
2	**Software**	102
2.1	Monitorprogramme	102
2.1.1	Befehle des Monitors	102
2.1.2	Funktionsweise des Monitorprogramms	103
2.1.3	Realisierung des Programms	106
2.1.4	Beispiel eines kommerziellen Monitorprogramms	116
2.2	Editor	122
2.2.1	Befehle des Editors	122
2.2.2	Funktionsweise des Editors	124
2.2.3	Verschiedene EDITOR Ausführungen	139
2.3	Assembler	139
2.3.1	Relocating Assembler	140
2.3.2	Linking Assembler	141
2.3.3	Makro Assembler	142
2.4	BASIC	144
2.4.1	RDK BASIC	144
2.4.1.1	BASIC Befehlssatz	145
2.4.1.2	Starten und Anpassen des Interpreters	157
2.4.2	12 K BASIC	161

2.5 Disassembler 164
2.5.1 Universal Disassembler in BASIC 164
2.6 Testhilfen für Software 170
2.6.1 Softwaretracer 172
2.6.1.1 Befehle des Tracers 172
2.6.1.2 Anpassen des Tracers an vorhandene Systeme 176
2.6.1.3 Funktionsweise des Tracers 178

3 Software, verschiedene Programme 179
3.1 Lader für das TDL Relocating Format 179
3.1.1 CPM Betriebssystem 179
3.1.2 Relocater 182
3.2 LIFE 192
3.3 Boolescher Formel-Analysator 199

Literaturverzeichnis 210

Bezugsquellenverzeichnis 211

Fachausdrücke – Glossar 212

Sachverzeichnis 219

1 Hardware

1.1 Z80 Aufbau und Befehlssatz

1.1.1 Z80 Hardware

Abb. 1.1.1-1 zeigt den schematischen Aufbau der Z80 CPU. Der Z80 ist ein 8-Bit Mikroprozessor der neueren Generation. Er benötigt nur eine 5 V Versorgungsspannung und einen einfachen Takt. Der Adreß- und Datenbus sind vollständig herausgeführt, so daß eine externe Demultiplexerlogik nicht nötig ist. *Abb. 1.1.1-2* zeigt die interne Registerstruktur des Z80. Er besitzt zwei Registersätze, die umgeschaltet werden können und von denen immer nur einer aktiv ist.

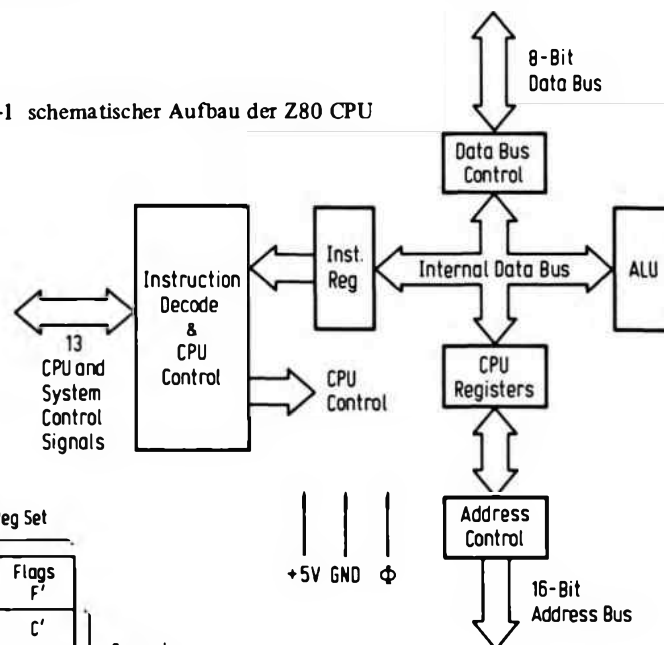


Abb. 1.1.1-1 schematischer Aufbau der Z80 CPU

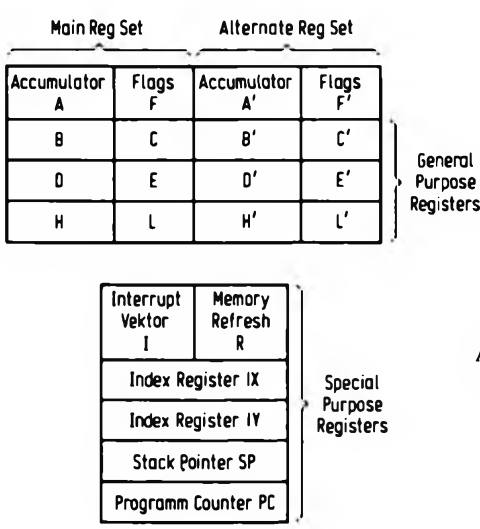


Abb. 1.1.1-2 interne Registerstruktur

Darunter befindet sich ein Registerpaar mit einem Akkumulator und Flagregister, die anderen drei Registerpaare BC, DE und HL sind für verschiedene Befehlsarten sowohl als Daten- als auch als Indexregister verwendbar.

Ferner besitzt der Z80 zwei 16-Bit Indexregister IX und IY, sowie einen Stackpointer SP. Außerdem ein Interruptvektorregister I und ein Refreshregister R, daß zum Anschluß von dynamischen Speichern verwendet werden kann.

Der Z80 ist aufwärtskompatibel zum 8080, das heißt Programme vom 8080 können auf dem Z80 laufen. (1) 8080 Befehlsbeschreibung. Leider ist dies nicht ausnahmslos richtig, denn der Z80 verwendet Paritätsflag etwas anders als der 8080, und Programme, die an der betreffenden Stelle diese Eigenschaft ausnützen, laufen auf dem Z80 anders ab.

Abb. 1.1.1-3 zeigt die Aufteilung des Flagregisters. So wird es auch bei der Ausführung des „PUSH PSW"-Befehls im Speicher abgelegt. Bit 7 stellt dabei die Vorzeichenstelle dar. Es ist identisch mit der Bedeutung beim 8080. Ebenfalls identisch ist die Bedeu-

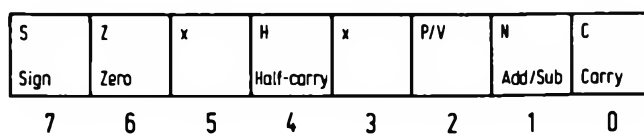


Abb. 1.1.1-3 Aufbau des Flagregisters

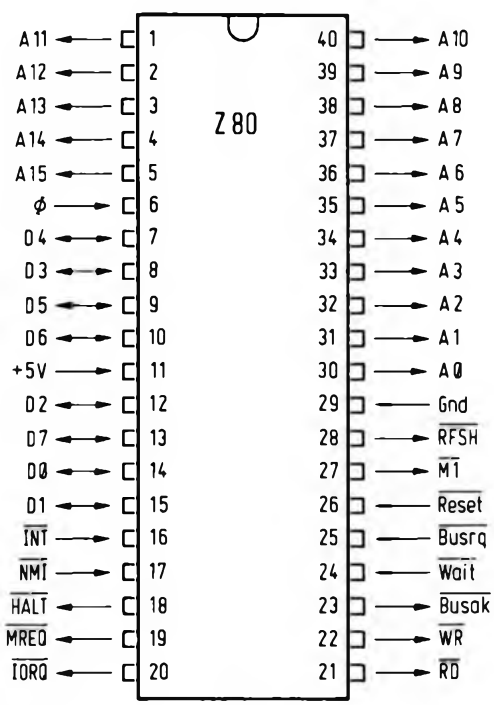


Abb. 1.1.1-4 PIN-Belegung des Z80

tung des Bit 6, daß den Nullzustand eines Registers anzeigt. Das gleiche gilt für das Übertragsbit aus der vierten Stelle, das hier aber Dank einer zusätzlichen Information mit Bit 1 auch der Dezimalkorrektur nach einer Subtraktion dienen kann.

Bit 2 besitzt auch eine abweichende Bedeutung. So wird es nicht nur zur Paritätsangabe verwendet, sondern bei manchen arithmetischen Befehlen dient es als Überlaufbit (overflow). Das Bit 0 ist wieder mit der Bedeutung als Übertrag (carry) identisch zu der 8080-Bedeutung. *Abb. 1.1.1-4* zeigt die PIN-Belegung des Z80.

A0 bis A15 sind die 16-Bit-Adresse für den Speicher und bei einem IO Befehl gelten die unteren 8 Bits, wobei die obere Hälfte des Adreßbusses dann den Inhalt des Akkumulators trägt (nur bei manchen Z-80-Befehlen).

D0 bis D7 Datenbus des Prozessors.

$\overline{M1}$ Mit diesem Signal wird angezeigt, wann ein Befehl von der CPU geholt wird. Das gilt auch für Mehrfach-OP-Codes wie beim Befehl INX X (Code: DD 23), bei dem dann zwei M1 Zyklen nötig sind.

$\overline{MREQ}$ Es wird angezeigt, daß eine gültige Speicheradresse an A0 bis A15 ansteht.

$\overline{IORQ}$ Die untere Hälfte des Adreßbusses enthält eine gültige Adresse für eine IO Operation. Tritt $\overline{IORQ}$ mit $\overline{M1}$ ein, so wird ein Interruptacknowlegde angezeigt.

$\overline{RD}$ Der Prozessor will Daten vom Speicher oder IO lesen.

$\overline{WR}$ Daten, die an dem Datenbus anliegen, sollen von der CPU in den Speicher oder IO geschrieben werden.

$\overline{RFSH}$ Auf der unteren Hälfte des Adreßbusses liegt mit 7 Bits eine Refreshadresse für dynamische Speicher. A7 ist 0 und nicht verwendet. Auf A8 bis A15 liegt der Inhalt des Interruptvektorregisters I.

$\overline{HALT}$ Die CPU hat einen HALT-Befehl (Code 76) ausgeführt. Die Operation kann nur nach einem NMI (non maskable interrupt), einem INT (interrupt) mit freigegebener Maske oder einem RESET wiederaufgenommen werden. Während der „Ausführung" des HALT-Befehls werden NOPs ausgeführt, so daß der Refresh von dynamischen Speichern trotzdem gewährleistet ist.

$\overline{WAIT}$ Der CPU wird angegeben, daß ein Speicher oder ein Periferiegerät Daten noch nicht bereithält oder verarbeiten kann. Damit können unterschiedlich schnelle Peripherieeinheiten mit der CPU synchronisiert werden. Während das $\overline{WAIT}$-Signal aktiv ist, ist ein Refresh unterbunden, so daß ein zu langes $\overline{WAIT}$-Signal einen Informationsverlust bei dynamischen Speichern verursachen kann.

$\overline{INT}$ Ein Interrupt wird nach Beenden eines laufenden Befehls angenommen, falls das interne Interruptenable-Flipflop im Zustand „ein" ist und kein $\overline{BUSRQ}$ (bus request) vorliegt. Dabei wird zum Beispiel bei einem

Blocktransfer, der sehr viele Maschinenzyklen lang sein kann, ein Interrupt auch während der Ausführung dieses Befehls angenommen und nach Rückkehr wird korrekt in der Ausführung dieses Befehls fortgefahren.

Der $\overline{INT}$-Eingang ist statisch und es ist darauf zu achten, daß nach einem Interruptacknowlegde ($\overline{M1}$ und $\overline{IORQ}$) die Anforderung wieder zurückgenommen wird. Der Z80 kennt drei verschiedene Arten auf diesen Interrupt zu reagieren, die per Software einstellbar sind.

$\overline{NMI}$ Dieser Eingang ist negativ flankengetriggert. Der Interrupt hat höchste Priorität und wird am Ende eines Befehls immer angenommen. Es wird dann auf die Speicherzelle mit der Adresse 66H (H bedeutet HEX, also sedezimal). Ein $\overline{BUSRQ}$ sowie ein $\overline{WAIT}$ kann den NMI davon abhalten sofort zu wirken.

$\overline{RESET}$ Der Programmzähler wird auf 0 gesetzt. Das Interruptenable-Flipflop wird in den Zustand „aus" (disable) gesetzt, das I-Register auf 0 und ebenfalls das R-Register. Es wird außerdem der Interruptmode 0 eingestellt, der das Verhalten von $\overline{INT}$ so steuert, daß es mit dem 8080-Interruptverhalten identisch ist.

$\overline{BUSRQ}$ Dieses Signal wird verwendet, um den Daten-, Adreß- und Steuerbus in den TRI-State Zustand zu überführen, sobald der laufende Maschinenzyklus beendet ist.

$\overline{BUSAK}$ Dieser Ausgang wird verwendet um die Annahme des BUSRQ-Signals anzuzeigen. Zu beachten ist, daß bei einem Busrequest natürlich kein Refresh von dynamischen Speichern durchgeführt wird und die externe Bussteuerung, durch die das $\overline{BUSRQ}$-Signal ausgelöst wurde, selbst dafür sorgen muß, daß ein Refresh aufrecht erhalten wird, falls BUSRQ zu lange dauert und dynamische Speicher im System verwendet wurden.

Φ Einfacher Takteingang, der TTL kompatibel ist und nur einen Widerstand von etwa 330 Ω nach + 5 V benötigt.

Timing:
Befehlsholphase:

Abb. 1.1.1-5 zeigt das Impulsdiagramm der Befehlsholphase (instruction op code fetch). Die Befehlsholphase wird auch als M1-Zyklus bezeichnet, da sie von einem Low-Pegel auf der M1-Leitung eingeleitet wird.

Das M1-Signal ist zur Steuerung des Speichers nicht erforderlich, es genügen die Signale RD und MREQ, die miteinander oder verknüpft ein Speicherfreigabesignal (chip select) ergeben, das über eine weitere Verknüpfung mit dem Adreßbereich direkt an ein EPROM geführt werden kann.

In der Zeit T3 bis T4 enthält der niederwertige Teil des Adreßbusses A0 bis A6 (nur 7 Bits) die Refreshadresse.

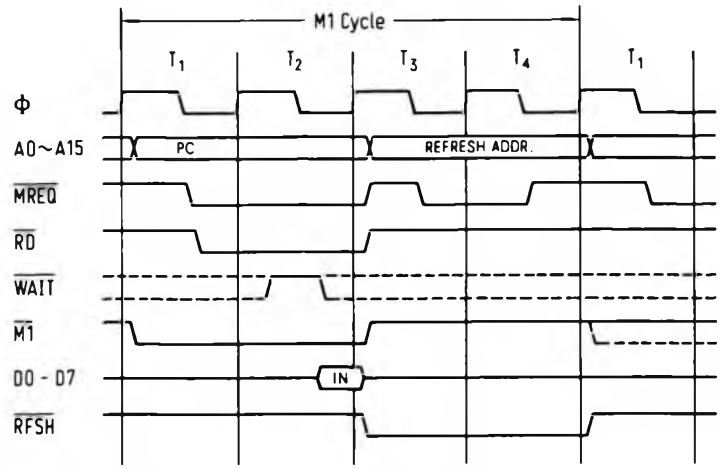


Abb. 1.1.1-5 Impulsdiagramm der Befehlsholphase

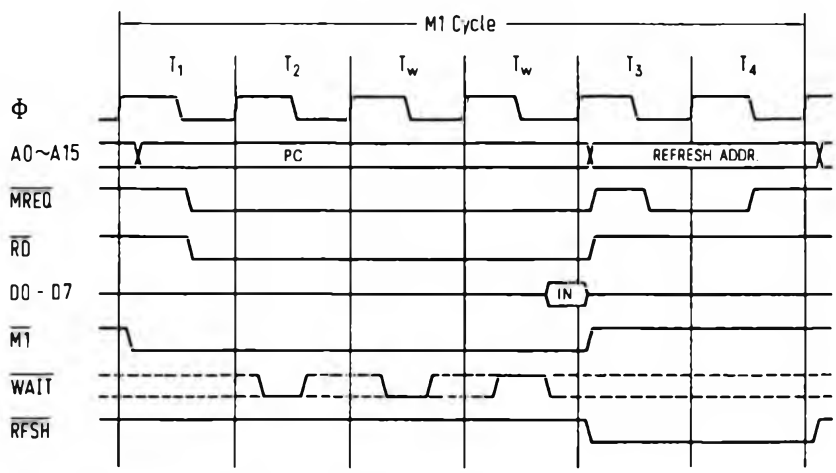


Abb. 1.1.1-6 Befehlsholphase mit Warte-Zyklen

Abb. 1.1.1-6 zeigt die Befehlsholphase, wenn zusätzliche Warte-Zyklen (wait cycles) eingefügt werden, um mit langsameren Speichern arbeiten zu können. Dabei wird der WAIT-Eingang mit der fallenden Flanke des Φ-Signals abgetastet.

RAM-Zugriffe:

Abb. 1.1.1-7 zeigt den Ablauf eines Lesevorgangs mit einem nachfolgenden Schreibzugriff. Durch die Signale RD und MREQ wird genau wie bei der Befehlsholphase ein

Lesevorgang eingeleitet. Bei Schreibzugriffen erscheinen die Signale WR und MREQ. Das $\overline{WR}$-Signal kann meist direkt dazu verwendet werden, um den R/$\overline{W}$-Eingang eines Speichers zu steuern.

Auch hier kann, wie in *Abb. 1.1.1-8* gezeigt, mit WAIT-Zyklen die Synchronisierung mit langsameren Speichern erreicht werden.

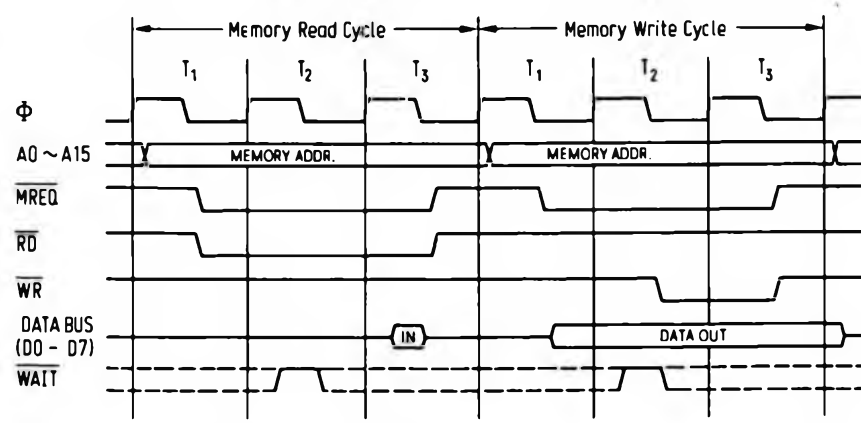


Abb. 1.1.1-7 Ablauf eines Lesevorgangs mit einem nachfolgenden Schreibzugriff

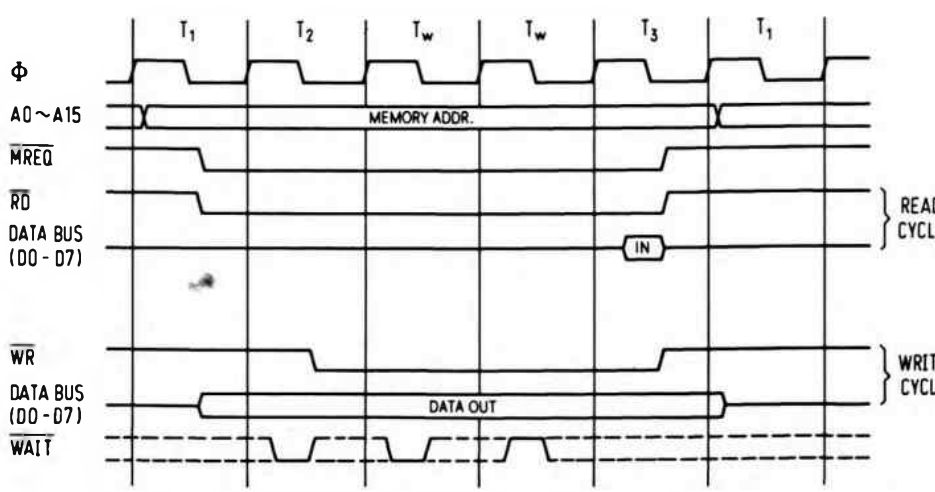


Abb. 1.1.1-8 Lese/Schreibzugriff mit Warte-Zyklen

Ein-, Ausgabe:

Bei Ein-, Ausgabevorgängen, dessen Impulsverlauf *Abb. 1.1.1-9* zeigt, wird immer automatisch ein WAIT-Zyklus von der CPU eingefügt. Die Selektierung der Peripheriebausteine kann bei einem Lesevorgang mit IORQ und RD erfolgen und bei einem Schreibzugriff mit IORQ und WR. *Abb. 1.1.1-10* zeigt den Ablauf mit weiteren WAIT-Zyklen.

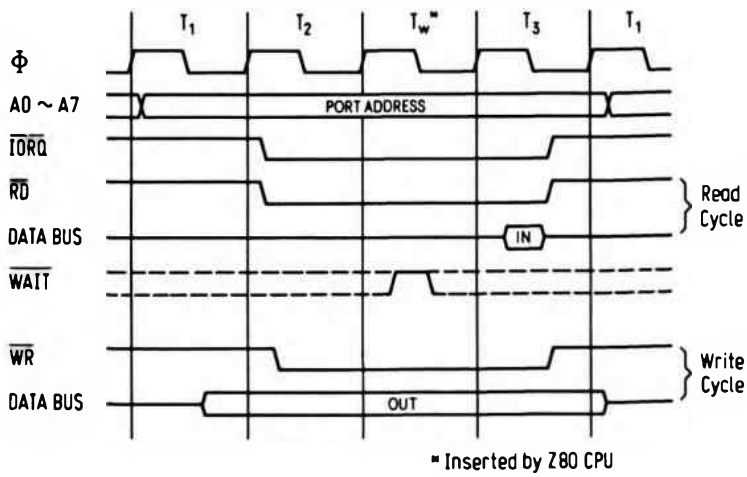


Abb. 1.1.1-9 Ein/Ausgabezugriffe der CPU

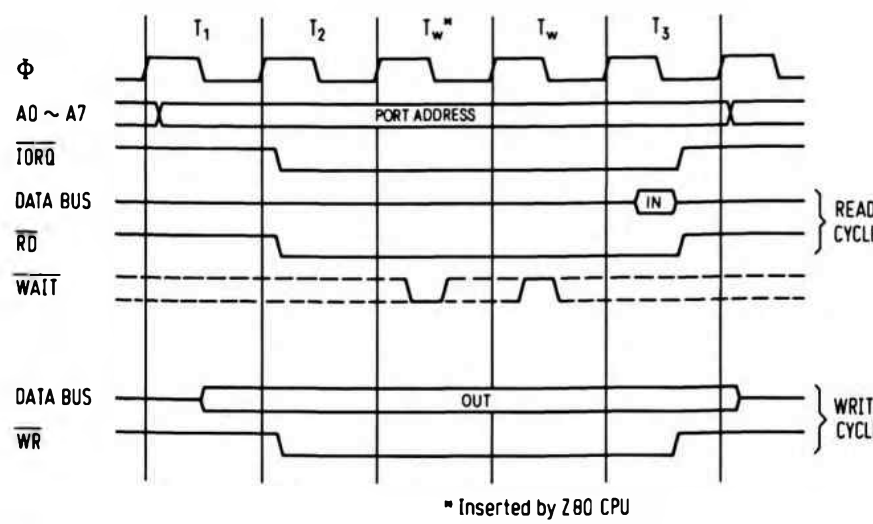


Abb. 1.1.1-10 Ein/Ausgabe mit Warte-Zyklen

Busanforderung (bus request):

Abb. 1.1.1-11 zeigt den Ablauf einer Busanforderung. Will ein externes DMA-Gerät (direct memory access) den Prozessorbus benützen, um Daten z.B. direkt von einem Periferiebaustein in den Speicher zu transportieren, so legt es die $\overline{\text{BUSRQ}}$-Leitung auf Low. Dieser Eingang wird von der CPU mit der steigenden Flanke des Taktsignals eines letzten Maschinenzyklus abgetastet. Danach wird der Bus freigegeben und dem DMA-Gerät durch einen Low-Pegel auf der $\overline{\text{BUSAK}}$-Leitung bestätigt.

Interrupt-Verarbeitung:

Abb. 1.1.1-12 zeigt das Impulsdiagramm für den nichtmaskierbaren Interrupt (NMI). Ein Puls auf der NMI-Leitung setzt ein internes Latch, das beim Ende eines jeden

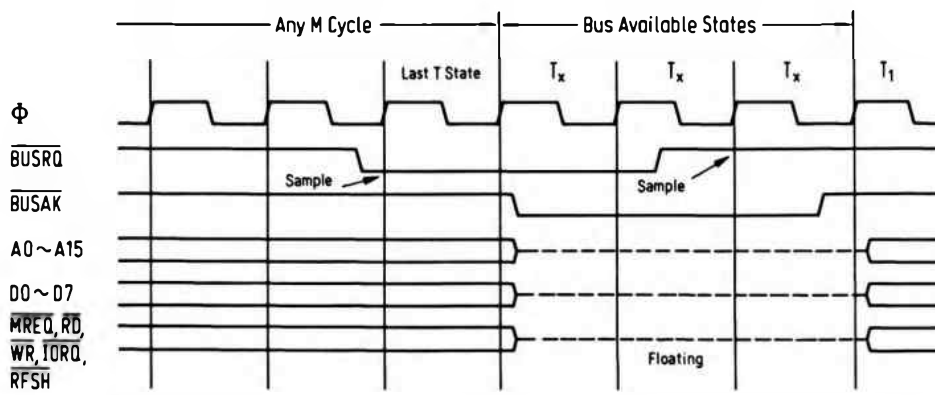


Abb. 1.1.1-11 Ablauf einer Busanforderung

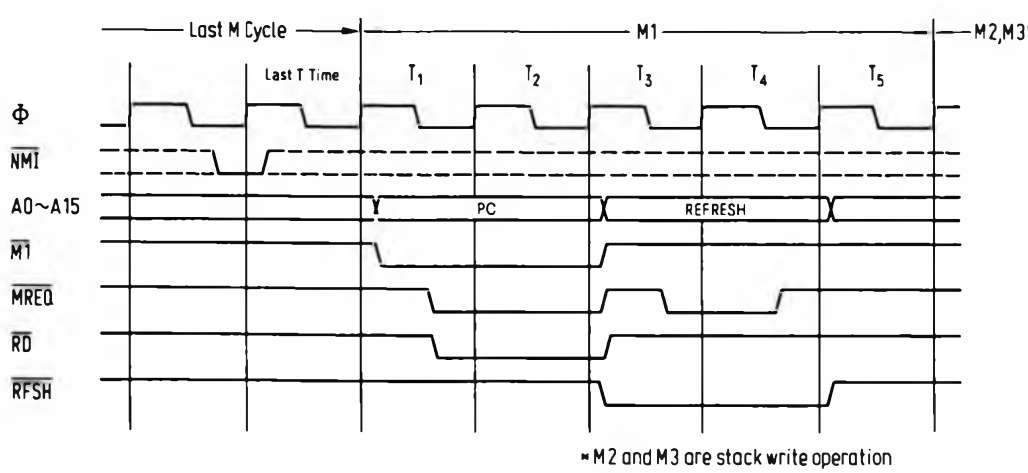


Abb. 1.1.1-12 Nichtmaskierbarer Interrupt (NMI)

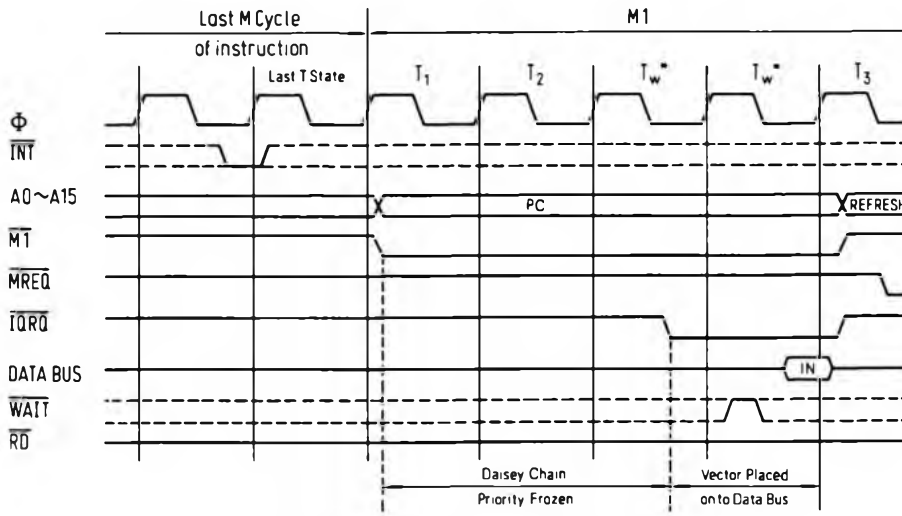


Abb. 1.1.1-13 Ablauf des Interrupt auf der INT-Leitung

Befehls abgefragt wird. Dies geschieht zur selben Zeit wie die Abfrage des INT-Eingangs, nur daß NMI die höhere Priorität besitzt. Der Prozessor speichert automatisch den alten Programmzähler in einem Stack (wie beim CALL-Befehl) und springt dann an die Speicherzelle mit der Adresse 66H. *Abb. 1.1.1-13* zeigt den Ablauf eines Interrupts auf der INT-Leitung. Der INT-Eingang ist statisch und wird nach Ende einer Befehlsausführung abgefragt. Falls das Interruptenable-Flipflop im Zustand „ein" war, und BUSRQ nicht aktiv ist, wird ein spezieller M1-Zyklus erzeugt. Dabei wird zu dem M1-Signal ein IORQ anstatt eines MREQ hinzugefügt, um dem externen Periferiegerät, das den Interrupt verursacht hat, zu zeigen, daß der Interrupt angenommen wurde. Dieses kann dann je nach eingestelltem Mode,z.B. einen Interruptvektor auf den Bus legen, den die CPU einliest.

Der Z80 besitzt drei Arten auf Interrupts zu reagieren. Im „Mode 0" arbeitet er wie der 8080. Es kann ein beliebiger Befehl auf den Datenbus gelegt werden, sobald der Interrupt angenommen wurde, dieser Befehl wird dann ausgeführt.

Meist wird ein RST- oder CALL-Befehl verwendet. Im „Mode 1" führt die CPU immer einen Restart auf die Adresse 38H aus, der die gleiche Wirkung wie die Ausführung des Befehls RST 7 hat.

Der „Mode 2" stellt die mächtigste Interruptbehandlung des Z80 dar. Mit einem einfachen 7-Bit Vektor ist es in Verbindung mit dem Register I möglich, einen Interrupt auf eine beliebige Stelle im Speicher durchzuführen.

Die höherwertigen 8 Bit werden vom I-Register genommen und die niederwertigen 7 Bit vom Datenbus. Dabei wird das Bit 0 als 0 angenommen. Damit wird eine Speicheradresse festgelegt, die nur auf geraden Speicherplätzen beginnen kann. Dort befindet sich eine Adreßtabelle und die CPU holt sich zwei weitere Bytes von dem Speicherplatz an und bildet damit eine neue Adresse, die nun als Startadresse des Interruptpro-

gramms interpretiert wird. Die Adresse ist in der Adreßtabelle beginnend mit dem niederwertigen Adreßteil abgelegt.

Abb. 1.1.1-14 zeigt den Interruptvorgang mit zusätzlichen WAIT-Zyklen.

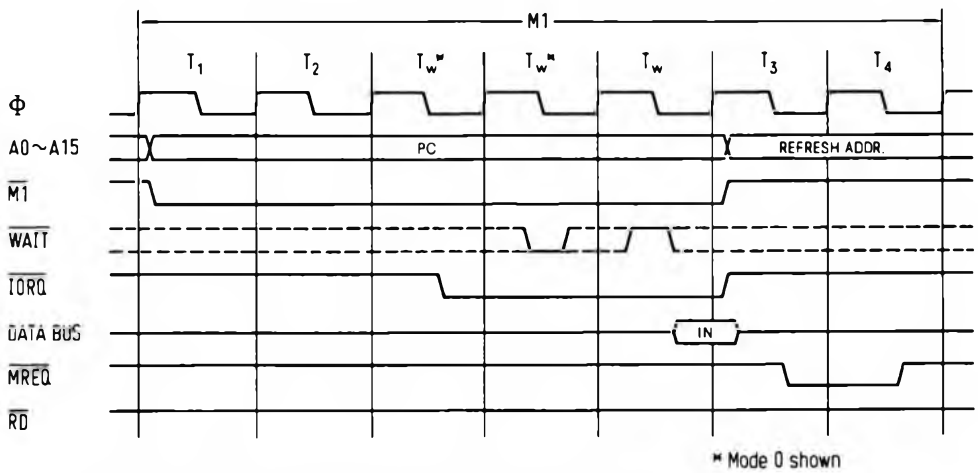


Abb. 1.1.1-14 Interrupt mit zusätzlichen Warte-Zyklen

HALT-Zustand:

Abb. 1.1.1-15 zeigt wie es möglich ist, einen durch den HALT-Befehl ausgelösten HALT-Zustand wieder zu verlassen.

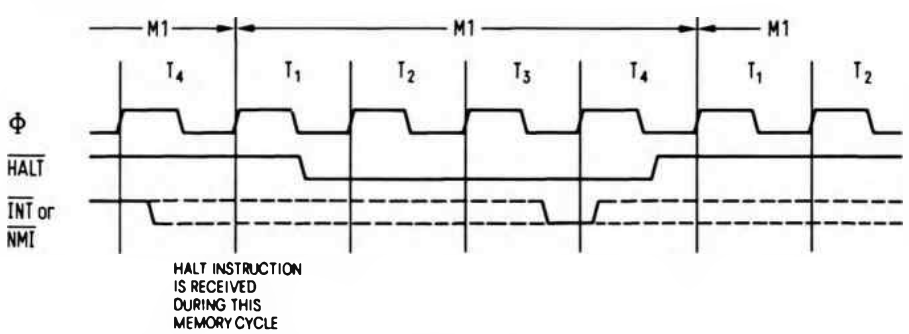


Abb. 1.1.1-15 Verlassen des HALT-Befehls

Schaltungstechnik

Quarz-Generator:

Abb. 1.1.1-16 zeigt eine einfache Schaltung zur Erzeugung eines 2-MHz-Taktes für den Z80.

Der Z80 wird in zwei Versionen angeboten. Die Standardversion ist für eine maximale Taktfrequenz von 2,5 MHz selektiert und die A-Version für 4 MHz.

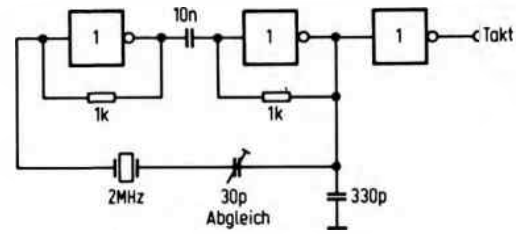


Abb. 1.1.1-16 Quarz-Taktgenerator

Anstelle des 2-MHz-Quarzes kann hier auch ein 2,5- oder 4-MHz-Quarz verwendet werden, doch dann werden Speicher benötigt, die eine Zugriffszeit kleiner als 450 ns besitzen, oder es müssen WAIT-Zyklen eingefügt werden.

Der Quarz-Generator ist mit Standard-TTL-Gatter bestückt (7404), die Frequenz kann mit einem 30-pF-Trimmer auf den genauen Wert von 2 MHz abgeglichen werden, falls dies z.B. für Echtzeitruhr o.ä. nötig sein sollte.

Der Aufbau des Quarzgenerators ist unkritisch, jedoch sollte auf eine kurze Leitungsführung geachtet werden. Als Quarz sollte ein Serienresonanztyp verwendet werden.

RESET-Logik:

Die RESET-Schaltung hat die Aufgabe, nach Einschalten der Versorgungsspannung dafür zu sorgen, daß der Prozessor in einen definierten Zustand gelangt, um mit der Abarbeitung des Programms zu beginnen. *Abb. 1.1.1-17* zeigt eine entsprechende Schaltung, die diese Aufgabe erfüllt. Nach Einschalten der Versorgungsspannung lädt sich der Kondensator C über den Widerstand R mit der Zeitkonstante RC auf.
Bei Erreichen der Schwellspannung wird das nachfolgende Gatter schalten. Der folgende Inverter N2 verbessert zusätzlich die Signalform, und das Signal gelangt an den Prozessor. Es empfiehlt sich als Gatter Ausführungsformen mit Schmittrigger zu verwenden, doch arbeitet die Schaltung auch mit den NAND-Gattern des Typs 74 LS 00. Dann darf jedoch der Ausgang von N1 nicht zur Steuerung weiterer Bausteine verwendet werden.

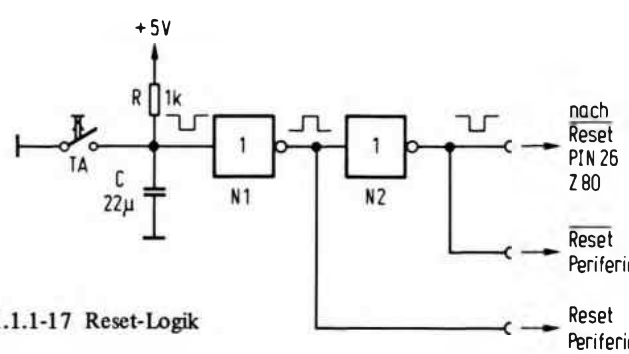


Abb. 1.1.1-17 Reset-Logik

Mit der Taste TA kann der RESET auch manuell ausgelöst werden. Enthält das Computersystem dynamische Speicher, die vom Prozessor refreshet werden, so führt ein manueller Tastendruck zum Informationsverlust und es ist nötig ein Monoflop nachzuschalten, daß den RESET-Impuls kurz hält.

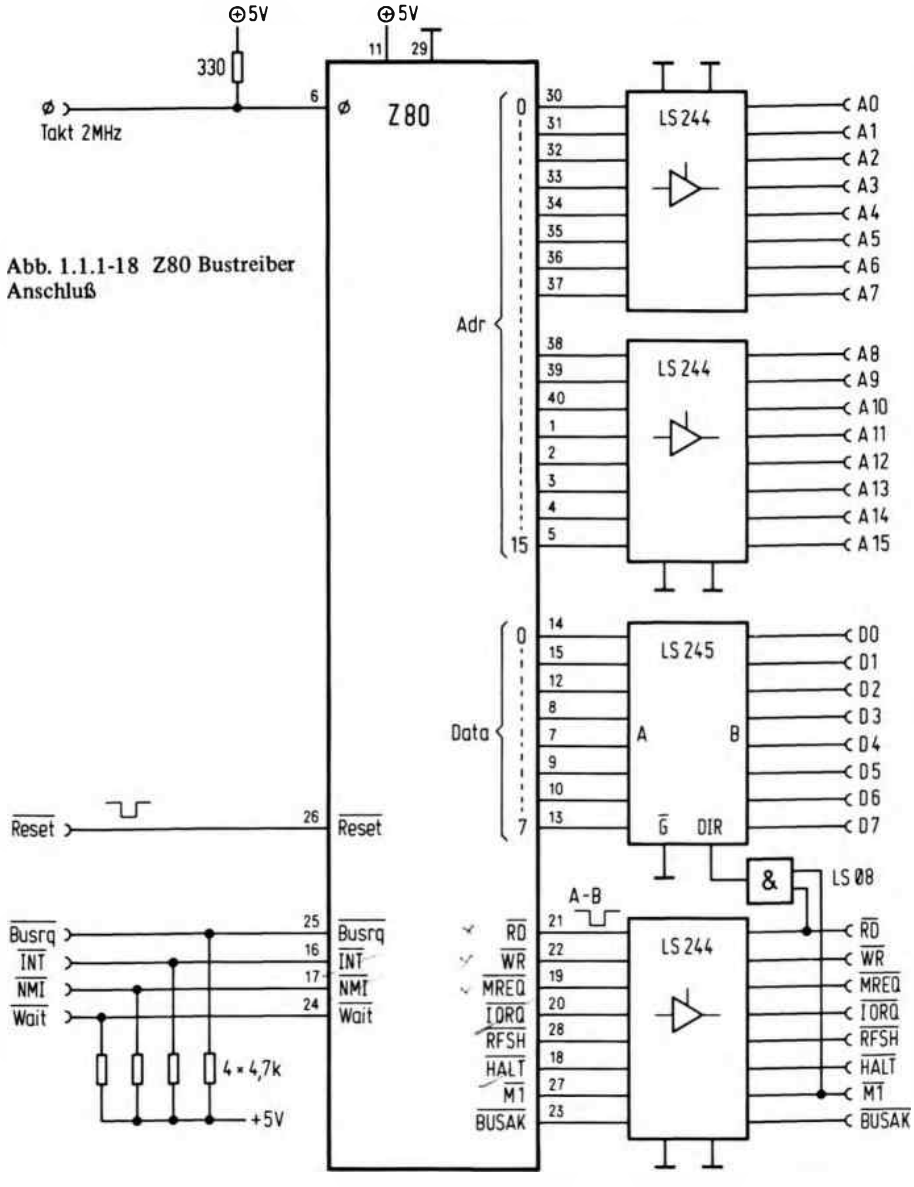


Abb. 1.1.1-18 Z80 Bustreiber Anschluß

Z80-Bustreiber-Anschluß:

Abb. 1.1.1-18 zeigt die dazugehörige Schaltung. Da der Z80 nur eine TTL-Last treiben kann, ist es bei größeren Systemen nötig, ihn zu puffern.

Der Adreßbus wird mit zwei ICs des Typs 74 LS 244 gepuffert. Dabei handelt es sich um Bustreiber, acht Stück in einem Gehäuse. Die TRI-State-enable-Eingänge sind hier nach Masse geschaltet, so daß der Adreßbus dauernd durchgeschaltet ist. Wird DMA-Betrieb gewünscht, so müssen die Selekteingänge mit BUSAK verknüpft werden.

Der Datenbus muß mit einem bidirektionalen Bustreiber gepuffert werden, dazu wird das IC 74 LS 245 verwendet. Es besitzt einen Selekteingang (enable) und einen Richtungseingang (direction). Der Selekteingang kann hier ebenfalls nach Masse geschaltet werden, so daß der Bustreiber dauernd aktiv ist. Der DIR (direktion)-Eingang wird so gesteuert, daß im Normalfall von der CPU weg auf den Systembus getrieben wird. Nur bei einem $\overline{RD}$ oder $\overline{M1}$-Signal, die über ODER/UND-Gatter verknüpft sind, wird die Richtung umgeschaltet. Ein Und-Gatter bewirkt hier eine Oder-Verknüpfung wegen der negativen Logik. Der M1-Ausgang muß verwendet werden, da bei einem Interruptacknowledge-Zyklus nur das M1-Signal erscheint und kein RD-Signal, obwohl Daten von der CPU gelesen werden sollen.

Der Steuerbus ist auch mit einem Bustreiber des Typs 74 LS 244 gepuffert.

Der Takteingang der CPU ist über einem 330-Ω-Widerstand mit + 5 V verbunden, um den Z80-Spezifikationen zu genügen. Da die CPU statisch ist, kann auch mit einer

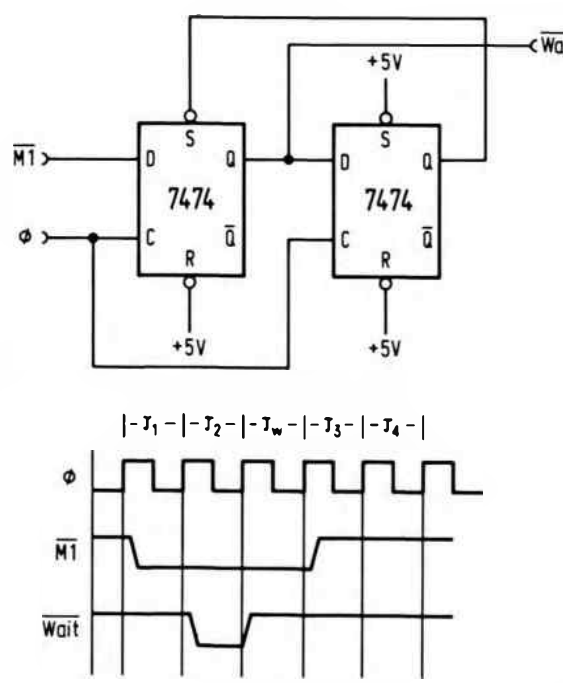


Abb. 1.1.1-19 M1-WAIT-Zyklus Erzeugung

beliebig kleinen Taktfrequenz gearbeitet werden, nur darf der Takt nicht mehr symmetrisch sein, da laut Datenblatt die Taktpulsbreite im Low-Zustand nur 2 ys lang sein darf. Die Verweilzeit im High-Zustand darf hingegen beliebig groß sein (wird aber nicht bei allen Herstellern garantiert). Die Reset-Leitung kann mit dem Ausgang des Gatters N2 in Abb. 1.1.1-17 verbunden werden.

Die Eingänge $\overline{\text{WAIT}}$, $\overline{\text{INT}}$, $\overline{\text{NMI}}$ und $\overline{\text{BUSRQ}}$ sind über Widerstände auf + 5 V gelegt, um auch im offenen Zustand ein Arbeiten der CPU zu ermöglichen.

WAIT-Schaltungen für den Z80:

Abb. 1.1.1-19 zeigt die Schaltung zur Erzeugung eines WAIT-Signals bei jedem M1-Zyklus. Diese Schaltung kann zum Beispiel bei einer Taktfrequenz von 2,5 MHz und Speichern mit einer Zugriffszeit von 450 ns verwendet werden. *Abb. 1.1.1-20* zeigt eine andere Variante einer WAIT-Schaltung. Diesmal wird ein WAIT-Signal bei jedem Speicherzugriff erzeugt, so daß noch langsamere Speicher verwendet werden können.

Steuerplatte für Einzelschritt:

Abb. 1.1.1-21 zeigt die Schaltung zur Erzeugung eines Φ Signals, wobei zusätzlich auf Einzelschritt umgeschaltet werden kann.

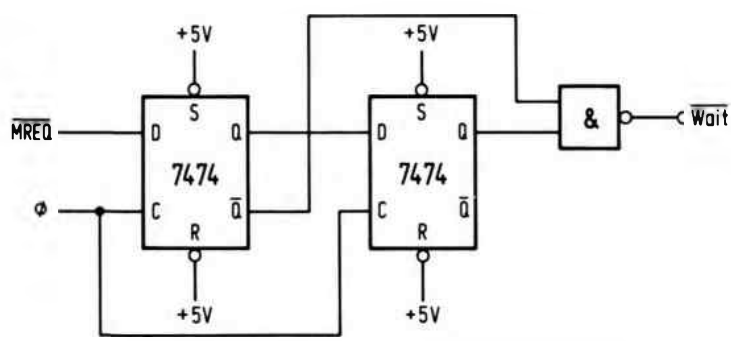
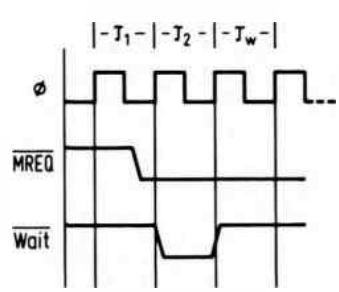


Abb. 1.1.1-20 WAIT-Zyklus bei Speicherzugriffen



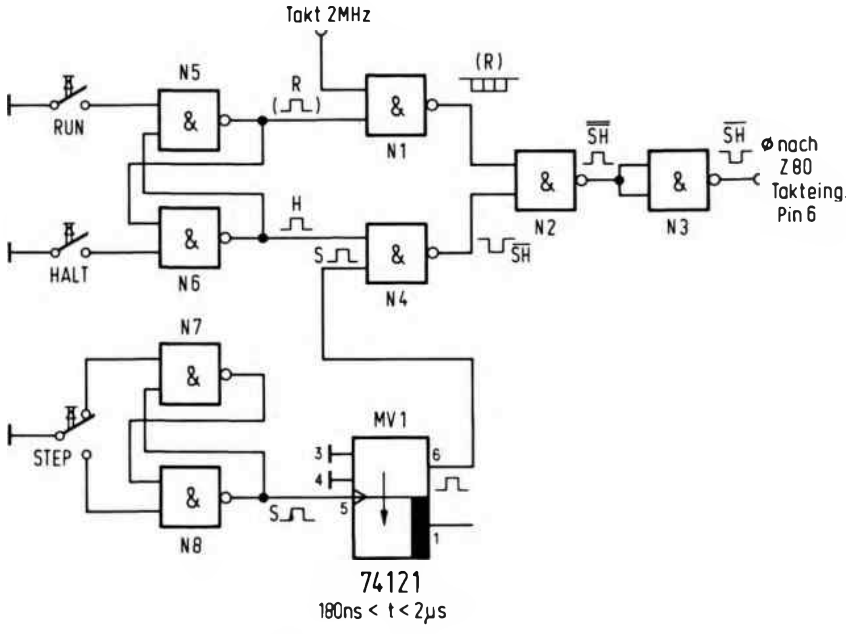


Abb. 1.1.1-21 Steuerplatte für Einzelschritt

Nach Betätigen der Taste RUN ist das Gatter N1 geöffnet und das 2-MHz-Signal kann an N2 gelangen. An dem anderen Eingang von N2 liegt ein High-Pegel, so daß der Takt über N3 an die CPU gelangen kann.

Wird die Taste HALT betätigt, so ist N1 gesperrt und N4 geöffnet, so daß der Ausgang des Monoflops MV1 an den Ausgang von N3 durchgeschaltet wird.
Bei Betätigen der STOP-Taste wird jeweils ein kurzer Impuls am Ausgang erzeugt.

Die Schaltung kann bei dynamischen Speichern nicht verwendet werden, da der Refresh im statischen Betrieb natürlich nicht laufend stattfindet.

Minimalsystem:

Abb. 1.1.1-22 zeigt eine Möglichkeit für die Realisierung eines Minimalsystems mit dem Z80.

Es kann für einfache Aufgaben wie Steuerung verwendet werden. Die CPU ist dazu mit einem Speicher von 1K Bytes EPROM zur Aufnahme des Programms und 1 K Byte Datenspeicher ausgerüstet. Als Eingabeport ist das IC 74 LS 244 verwendet und als Ausgabeport ein Latch des Typs 74 LS 273. Die Ports werden direkt mit den Signalen IORD bzw. IOWR gesteuert, ohne zusätzliche Verknüpfung mit einer Adresse, da hier für jede Datenrichtung nur ein Port vorhanden ist.

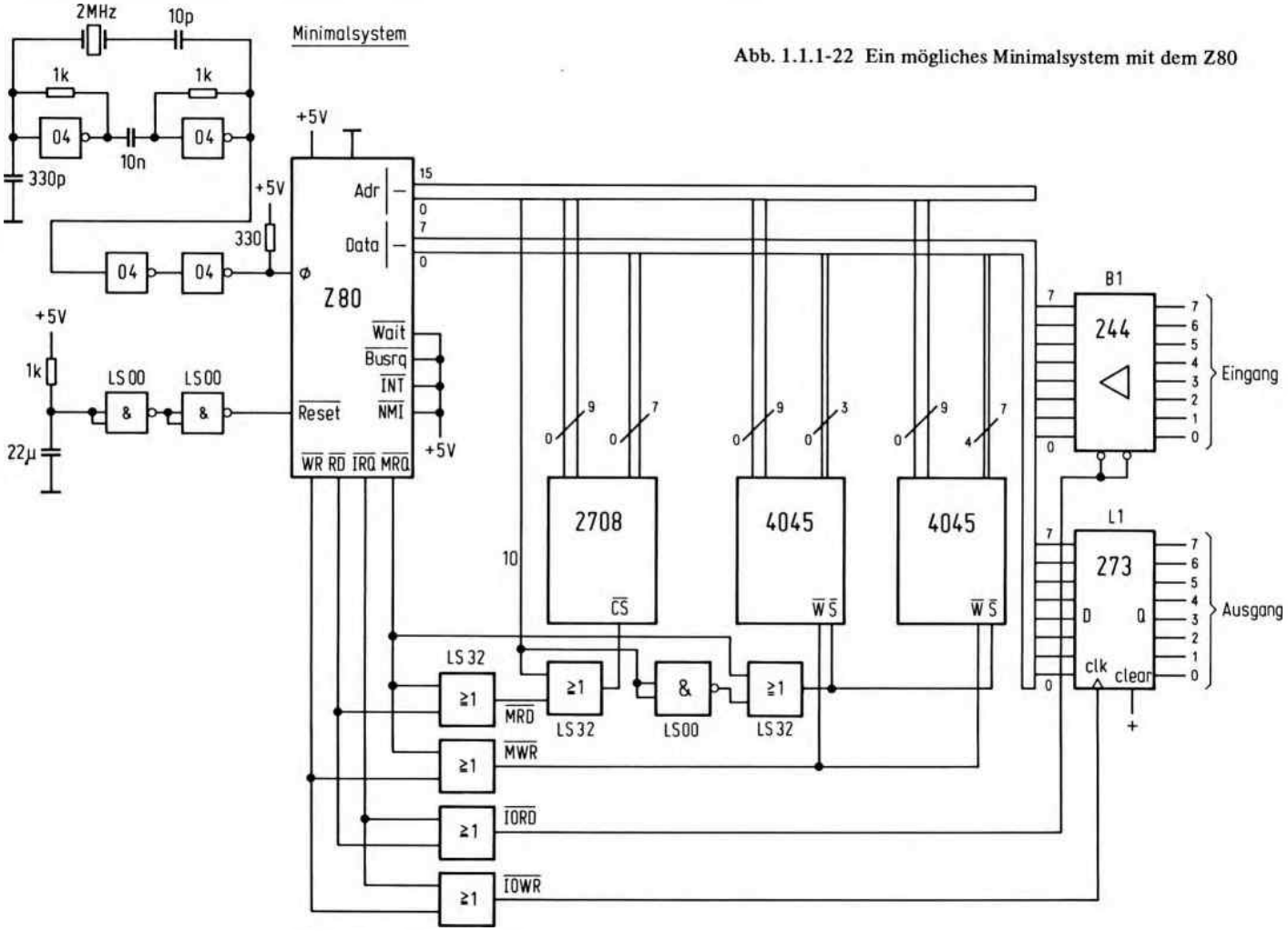


Abb. 1.1.1-22 Ein mögliches Minimalsystem mit dem Z80

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LD BC dddd	LD(BC) ,A	INC BC	INC B	DEC B	LD B, dd	RLCA	EX AF,AF'	ADD HL,BC	LDA, (BC)	DEC BC	INC C	DEC C	LD C dd	RRCA
1	DJNZ disp	LD DE dddd	LD(DE) ,A	INC DE	INC D	DEC D	LD D, dd	RLA	JR, disp	ADD HL,DE	LDA, (DE)	DEC DE	INC E	DEC E	LD E, dd	RRA
2	JR NZ, disp	LD HL, dddd	LD(adr) ,HL	INC HL	INC H	DEC H	LD H, dd	DAA	JR Z, disp	ADD HL,HL	LD HL ,(adr)	DEC HL	INC L	DEC L	LD L, dd	CPL
3	JRNC, disp	LD SP, dddd	LD(adr) ,A	INC SP	INC (HL)	DEC (HL)	LD(HL) ,dd	SCF	JR C, disp	ADD HL,SP	LD HL (adr)	DEC SP	INC A	DEC A	LD A, dd	CCF
4	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
5	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
6	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
7	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD (HL)	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC (HL)	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB (HL)	SUB A	SBC B	SBC C	SBC D	SBC E	SBC H	SBC L	SBC (HL)	SBC A
A	AND B	AND C	AND D	AND E	AND H	AND L	AND (HL)	AND A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR (HL)	XOR A
B	OR B	OR C	OR D	OR E	OR H	OR L	OR (HL)	OR A	CP B	CP C	CP D	CP E	CP H	CP L	CP (HL)	CP A
C	RET NZ	POP BC	JP NZ,adr	JP adr	CALL NZ,adr	PUSH BC	ADD A,dd	RST 0	RET Z	RET	JP Z,adr		CALL Z,adr	CALL adr	ADC A,dd	RST 8
D	RET NC	POP DE	JP NC,adr	OUT port,A	CALL NC,(adr)	PUSH DE	SUB dd	RST 10H	RET C	EXX	JP C,adr	IN A,port	CALL C,adr		SBC A,dd	RST 10H
E	RET PO	POP HL	JP PO,adr	EX (SP),HL	CALL PO,adr	PUSH HL	AND dd	RST 20H	RET PE	JP (HL)	JP PE,adr	EX DE,HL	CALL PE,adr		XOR dd	RST 28H
F	RET P	POP AF	JP P,adr	DI	CALL P,adr	PUSH AF	OR dd	RST 30H	RET M	LD (SP),HL	JP M,adr	EI	CALL M,adr		CP dd	RST 38H

Abb. 1.1.2-1 Befehlstabelle Z80, 1-Byte-Opcode

1.1.2 Software

Der Z80 ist wie schon erwähnt, aufwärtskompatibel zum 8080, deshalb sei hier nochmals auf das Buch „Mikrocomputersysteme" (1) verwiesen, in dem der 8080-Befehlssatz ausführlich beschrieben wird.

Der gesamte Befehlssatz des Z80 ist so umfangreich, daß mit der genauen Beschreibung ein Buch allein gefüllt werden kann. Daher weise ich hier auf das Assemblerhandbuch von ZILOG (Kontron) hin, das in deutscher Sprache erhältlich ist und eine ausführliche Beschreibung enthält (15). *Abb. 1.1.2-1* zeigt alle möglichen 1-Byte-Opcodebefehle, die natürlich auch aus mehreren Bytes bestehen können, wobei nur noch Operanden folgen.

Der Z80 besitzt aber weit mehr als nur 256 Befehlscodes, die sich aus kombinatorischen Gründen bei 8-Bit Befehlsbreite ergeben, da er Umschalt-Opcodes besitzt.
Mit den Bytes DDH, FDH, EDH, CBH und DDCBH sowie FDCBH gelangt man in neue Tabellen, die wieder 256 Kombinationen enthalten, von denen aber i.a. sehr wenige ausgenützt sind.

Abb. 1.1.2-2 zeigt alle bekannten Befehle, die mit EDH beginnen.
In *Abb. 1.1.2-3* sind die Befehle dargestellt, die mit DDH oder FDH beginnen. Der vorgeschaltete Opcode DDH (FDH) bewirkt im Prozessor nur eine Umschaltung des Registerpaars HL auf IX (bzw. IY). Der nächste folgende Befehl wird dann anstatt mit HL mit IX bzw. IY ausgeführt. Gegebenenfalls wird noch eine zusätzliche Distanzadresse (displacement) geholt.

Diese Regel gilt auch für die nicht im Assemblerhandbuch aufgeführten Codekombinationen. Wird zum Beispiel der Code DD 68 H in den Speicher eingegeben und ausgeführt, so wird der Inhalt des Registers B in den niederwertigen Teil des IX-Registers gebracht. 68 H ist dabei der Code für den Befehl MOV L,B (LD L,B).

Diese „Pseudobefehle" sollten aber möglichst nicht verwendet werden, da es nicht garantiert ist, daß bei späteren Z80-Versionen die Befehle noch so ablaufen.

Abb. 1.1.2-4 zeigt die Befehlsgruppe, der der Code CBH vorgeschaltet ist. In dieser Gruppe sind hauptsächlich Bitmanipulationsbefehle untergebracht.

Auffallend ist die Lücke in der Spalte 3x. Die dort vorhandenen Pseudobefehle bewirken eine Schiebeoperation, die ein Linksschieben verursacht. Das niederwertigste Bit wird mit 1 aufgefüllt.

Der Befehl CB 31 H wirkt zum Beispiel auf das C-Register. Enthält dieses zu Beginn den Wert 04, so beinhaltet es nach dem ersten Schiebebefehl den Wert 09 und nach einem weiteren Schiebevorgang den Wert 13 H.

Abb. 1.1.2-5 zeigt die 3-Byte-Opcodebefehle, die durch Vorschalten der Umschaltcodes DDH bzw. FDH zustande kommen, bei einem nachfolgenden CBH-Code.
Zilog definierte für den Z80 eigene Assemblerabkürzungen, die logischer aufgebaut sind als die 8080-Mnemonics. Manche Firmen haben aber ursprünglich für den 8080 entwickelte Assembler einfach erweitert und die alten 8080-Mnemonics um die Z80-Befehle erweitert. Sie mußten aber dann eigene Mnemonics für den Z80-Teil verwenden.

ED-Tabelle

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4	IN B	OUT B	SBC BC	LD (adr),BC	NEG	RETN	IM Ø	LD I,A	IN C	OUT C	ADC BC	LD BC,(adr)		RETI		LD R,A
5	IN D	OUT D	SBC DE	LD (adr),DE			IM 1	LD A,I	IN E	OUT E	ADC DE	LD DE,(adr)			IM 2	LD A,R
6	IN H	OUT H	SBC HL					RRD	IN L	OUT L	ADC HL					RLD
7			SBC SP	LD (adr),SP					IN A	OUT A	ADC SP	LD SP,(nn)				
8																
9																
A	LDI	CPI	INI	OUTI					LDD	CPD	IND	OUTD				
B	LDIR	CPIR	INIR	OTIR					LDDR	CPDR	INDR	OTDR				
C																
D																
E																
F																

Abb. 1.1.2-2 Befehle mit führendem EDH als Code

	DD	FD
(II) =	(IX+d)	(IY+d)
II =	IX	IY
(IR) =	(IX)	(IY)

	1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0										ADD II,BC						
1										ADD II,DE						
2		LD II, dddd	LD (adr),II	INC II						ADD II,II	LD II, (adr)	DEC II				
3					INC (II)	DEC (II)	LD (II),dd			ADD II,SP						
4							LD B,(II)								LD C,(II)	
5							LD D,(II)								LD E,(II)	
6							LD H,(II)								LD L,(II)	
7	LD (II),B	LD (II),C	LD (II),D	LD (II),E	LD (II),H	LD (II),L		LD (II),A							LD A,(II)	
8							ADD (II)								ADC (II)	
9							SUB (II)								SBC (II)	
A							AND (II)								XOR (II)	
B							OR (II)								CP (II)	
C																
D																
E		POP II		EX (SP),II		PUSH II				JP (IR)						
F																

Abb. 1.1.2-3 Befehle mit führendem DDH oder FDH

CB-Tabelle

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC (HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
1	RL B	RL C	RL D	RL E	RL H	RL L	RL (HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
2	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA (HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
3									SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
4	BIT 0,B	BIT 0,C	BIT 0,D	BIT 0,E	BIT 0,H	BIT 0,L	BIT 0,(HL)	BIT 0,A	BIT 1,B	BIT 1,C	BIT 1,D	BIT 1,E	BIT 1,H	BIT 1,L	BIT 1,(HL)	BIT 1,A
5	BIT 2,B	BIT 2,C	BIT 2,D	BIT 2,E	BIT 2,H	BIT 2,L	BIT 2,(HL)	BIT 2,A	BIT 3,B	BIT 3,C	BIT 3,D	BIT 3,E	BIT 3,H	BIT 3,L	BIT 3,(HL)	BIT 3,A
6	BIT 4,B	BIT 4,C	BIT 4,D	BIT 4,E	BIT 4,H	BIT 4,L	BIT 4,(HL)	BIT 4,A	BIT 5,B	BIT 5,C	BIT 5,D	BIT 5,E	BIT 5,H	BIT 5,L	BIT 5,(HL)	BIT 5,A
7	BIT 6,B	BIT 6,C	BIT 6,D	BIT 6,E	BIT 6,H	BIT 6,L	BIT 6,(HL)	BIT 6,A	BIT 7,B	BIT 7,C	BIT 7,D	BIT 7,E	BIT 7,H	BIT 7,L	BIT (7,(HL	BIT 7,A
8	RES 0,B	RES 0,C	RES 0,D	RES 0,E	RES 0,H	RES 0,L	RES 0,(HL	RES 0,A	RES 1,B	RES 1,C	RES 1,D	RES 1,E	RES 1,H	RES 1,L	RES 1,(HL)	RES 1,A
9	RES 2,B	RES 2,C	RES 2,D	RES 2,E	RES 2,H	RES 2,L	RES 2,(HL)	RES 2,A	RES 3,B	RES 3,C	RES 3,D	RES 3,E	RES 3,H	RES 3,L	RES 3,(HL)	RES 3,A
A	RES 4,B	RES 4,C	RES 4,D	RES 4,E	RES 4,H	RES 4,L	RES 4,(HL)	RES 4,A	RES 5,B	RES 5,C	RES 5,D	RES 5,E	RES 5,H	RES 5,L	RES 5,(HL)	RES 5,A
B	RES 6,B	RES 6,C	RES 6,D	RES 6,E	RES 6,H	RES 6,L	RES 6,(HL)	RES 6,A	RES 7,B	RES 7,C	RES 7,D	RES 7,E	RES 7,H	RES 7,L	RES 7,(HL)	RES 7,A
C	SET 0,B	SET 0,C	SET 0,D	SET 0,E	SET 0,H	SET 0,L	SET 0,(HL)	SET 0,A	SET 1,B	SET 1,C	SET 1,D	SET 1,E	SET 1,H	SET 1,L	SET 1,(HL)	SET 1,A
D	SET 2,B	SET 2,C	SET 2,D	SET 2,E	SET 2,H	SET 2,L	SET 2,(HL)	SET 2,A	SET 3,B	SET 3,C	SET 3,D	SET 3,E	SET 3,H	SET 3,L	SET 3,(HL)	SET 3,A
E	SET 4,B	SET 4,C	SET 4,D	SET 4,E	SET 4,H	SET 4,L	SET 4,(HL)	SET 4,A	SET 5,B	SET 5,C	SET 5,D	SET 5,E	SET 5,H	SET 5,L	SET 5,(HL)	SET 5,A
F	SET 6,B	SET 6,C	SET 6,D	SET 6,E	SET 6,H	SET 6,L	SET 6,(HL)	SET 6,A	SET 7,B	SET 7,C	SET 7,D	SET 7,E	SET 7,H	SET 7,L	SET 7,(HL)	SET 7,A

Abb. 1.1.2-4 Befehle mit führendem CBH

DD / FD CB
n = (IX+d) (IY+d)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0							RLC n								RRC n	
1							RL n								RR n	
2							SLA n								SRA n	
3															SRL n	
4							BIT 0,n								BIT 1,n	
5							BIT 2,n								BIT 3,n	
6							BIT 4,n								BIT 5,n	
7							BIT 6,n								BIT 7,n	
8							RES 0,n								RES 1,n	
9							RES 2,n								RES 3,n	
A							RES 4,n								RES 5,n	
B							RES 6,n								RES 7,n	
C							SET 0,n								SET 1,n	
D							SET 2,n								SET 3,n	
E							SET 4,n								SET 5,n	
F							SET 6,n								SET 7,n	

Abb. 1.1.2-5 3-Byte-Opcode Befehle mit DDH/FDH CBH

Abb. 1.1.2-6 zeigt eine Gegenüberstellung von drei gebräuchlichen Z80 Assemblermnemonics. In der Mitte sind die Zilog-Definitionen dargestellt, auf der linken Seite die Version, wie sie vom Makrolibrary des CPM 8080 MAC Assemblers zur Verfügung stehen, rechts die Definitionen von TDL, wie sie auch im Buch verwendet werden. Dabei ist die Bezeichnung IX, IY bei der Version des Assemblers im Buch mit X und Y dargestellt.

Z-80 Macro Library Documentation

I.

The purpose of this library is to enable the assembly of the Z-80 instruction set on a CP/M sytem using the CP/M MAC macro assembler.

This library is invoked with the pseudo-op:

" MACLIB Z80 "

II.

The following symbols and notations are used in the individual macro descriptions;

r - Any of the 8 bit registers: A, B, C, D, E, H, L, or M
rr - Any of the 16 bit register pairs: BC, DE, HL, or SP
nn - 8 bit immediate data (0 through 255)
d - 8 bit signed displacment (-128 through +127)
nnnn - 16 bit address or immediate data (0 through 65535)
b - bit number (0-7, 7 is most significant, 0 is least)
addr - 16 bit address within PC+127 through PC-128
m(zzz) - Memory at address "zzz"

III.

MACLIB ver		Zilog ver		TDL ver	
LDX	r,d	LD	r,(IX+d)	MOV	r,d(IX)
Load register from indexed memory (with IX)					
LDY	r,d	LD	r,(IY+d)	MOV	r,d(IY)
Load register from indexed memory (with IY)					
STX	r,d	LD	(IX+d),r	MOV	d(IX),r
Store register to indexed memory (with IX)					
STY	r,d	LD	(IY+d),r	MOV	d(IY),r
Store register to indexed memory (with IY)					
MVIX	nn,d	LD	(IX+d),nn	MVI	d(IX)
Move immediate to indexed memory (with IX)					
MVIY	nn,d	LD	(IY+d),nn	MVI	d(IY)
Move immediate to indexed memory (with IY)					
LDAI		LD	A,I	LDAI	
Move I to A					
LDAR		LD	A,R	LDAR	
Move R to A					
STAI		LD	I,A	STAI	
Move A to I					
STAR		LD	R,A	STAR	
Move A to R					
LXIX	nnnn	LD	IX,nnnn	LXI	IX,nnnn
Load IX immediate (16 bits)					
LXIY	nnnn	LD	IY,nnnn	LXI	IY,nnnn
Load IY immediate (16 bits)					

Abb. 1.1.2-6 Gegenüberstellung von Assemblermnemonics für den Z80

```
LBCD    nnnn            LD      BC,(nnnn)       LBCD    nnnn
    Load BC direct (from memory at nnnn)

LDED    nnnn            LD      DE,(nnnn)       LDED    nnnn
    Load DE direct

LSPD    nnnn            LD      SP,(nnnn)       LSPD    nnnn
    Load SP direct

LIXD    nnnn            LD      IX,(nnnn)       LIXD    nnnn
    Load IX direct

LIYD    nnnn            LD      IY,(nnnn)       LIYD    nnnn
    Load IY direct

SBCD    nnnn            LD      (nnnn),BC       SBCD    nnnn
    Store BC direct (to memory at nnnn)

SDED    nnnn            LD      (nnnn),DE       SDED    nnnn
    Store DE direct

SSPD    nnnn            LD      (nnnn),SP       SSPD    nnnn
    Store SP direct

SIXD    nnnn            LD      (nnnn),IX       SIXD    nnnn
    Store IX direct

SIYD    nnnn            LD      (nnnn),IY       SIYD    nnnn
    Store IY direct

SPIX                    LD      SP,IX           SPIX
    Copy IX to the SP

SPIY                    LD      SP,IY           SPIY
    Copy IY to the SP

PUSHIX                  PUSH    IX              PUSH    IX
    Push IX into the stack

PUSHIY                  PUSH    IY              PUSH    IY
    Push IY into the stack

POPIX                   POP     IX              POP     IX
    Pop IX from the stack

POPIY                   POP     IY              POP     IY
    Pop IY from the stack

EXAF                    EX      AF,AF'          EXAF
    Exchange AF and the alternate, AF'

EXX                     EXX                     EXX
    Exchange BC DE HL with BC' DE' HL'

XTIX                    EX      (SP),IX         XTIX
    Exchange IX with the top of the stack

XTIY                    EX      (SP),IY         XTIY
    Exchange IY with the top of the stack

LDI                     LDI                     LDI
    Move m(HL) to m(DE), increment DE and HL, decrement BC
```

```
LDIR                    LDIR                    LDIR
    Repeat LDI until BC = 0

LDD                     LDD                     LDD
    Move m(HL) to m(DE), decrement HL, DE, and BC

LDDR                    LDDR                    LDDR
    Repeat LDD until BC = 0

CCI                     CPI                     CCI
    Compare A with m(HL), increment HL, decrement BC

CCIR                    CPIR                    CCIR
    Repeat CCI until BC = 0  or  A = m(HL)

CCD                     CPD                     CCD
    Compare A with m(HL), decrement HL and BC

CCDR                    CPDR                    CCDR
    Repeat CCD until BC = 0 or A = m(HL)

ADDX    d               ADD     (IX+d)          ADD     d(IX)
    Indexed add to A

ADDY    d               ADD     (IY+d)          ADD     d(IY)
    Indexed add to A

ADCX    d               ADC     (IX+d)          ADC     d(IX)
    Indexed add with carry

ADCY    d               ADC     (IY+d)          ADC     d(IY)
    Indexed add with carry

SUBX    d               SUB     (IX+d)          SUB     d(IX)
    Indexed subtract

SUBY    d               SUB     (IY+d)          SUB     d(IY)
    Indexed Subtract

SBCX    d               SBC     (IX+d)          SBB     d(IX)
    Indexed subtract with "borrow"

SBCY    d               SBC     (IY+d)          SBB     d(IY)
    Indexed subtract with borrow

ANDX    d               AND     (IX+d)          ANA     d(IX)
    Indexed logical and

ANDY    d               AND     (IY+d)          ANA     d(IY)
    Indexed logical and

XORX    d               XOR     (IX+d)          XRA     d(IX)
    Indexed logical exclusive or

XORY    d               XOR     (IY+d)          XRA     d(IY)
    Indexed logical exclusive or

ORX     d               OR      (IX+d)          ORA     d(IX)
    Indexed logical or

ORY     d               OR      (IY+d)          ORA     d(IY)
    Indexed logical exclusive or
```

zu Abb. 1.1.2-6

```
CMPX    d               CP      (IX+d)          CMP     d(IX)
    Indexed compare

CMPY    d               CP      (IY+d)          CMP     d(IY)
    Index compare

INRX    d               INC     (IX+d)          INR     d(IX)
    Increment memory at m(IX+d)

INRY    d               INC     (IY+d)          INR     d(IY)
    Increment memory at m(IY+d)

DCRX    d               INC     (IX+d)          INR     d(IX)
    Decrement memory at m(IX+d)

DCRY    d               DEC     (IY+d)          DCR     d(IY)
    Decrement memory at m(IX+d)

NEG                     NEG                     NEG
    Negate A (two's complement)

IM0                     IM0                     IM0
    Set interrupt mode 0

IM1                     IM1                     IM1
    Set interrupt mode 1

IM2                     IM2                     IM2
    Set interrupt mode 2

DADC    rr              ADC     HL,rr           DADC    rr
    Add with carry rr to HL

DSBC    rr              SBC     HL,rr           DSBC    rr
    Subtract with "borrow" rr from HL

DADX    rr              ADD     IX,rr           DADX    rr
    Add rr to IX (rr may be BC, DE, SP, IX)

DADY    rr              ADD     IY,rr           DADY    rr
    Add rr to IY (rr may be BC, DE, SP, IY)

INXIX                   INC     IX              INX     IX
    Increment IX

INXIY                   INC     IY              INX     IY
    Increment IY

DCXIX                   DEC     IX              DCX     IX
    Decrement IX

DCXIY                   DEC     IY              DCX     IY
    Decrement IY

BIT     b,r             BIT     b,r             BIT     b,r
    Test bit b in register r

SETB    b,r             SET     b,r             SET     b,r
    Set bit b in register r
```

```
RES     b,r             RES     b,r             RES     b,r
    Reset bit b in register r

BITX    b,d             BIT     b,(IX+d)        BIT     b,d(IX)
    Test bit b in memory at m(IX+d)

BITY    b,d             BIT     b,(IY+d)        BIT     b,d(IY)
    Test bit b in memory at m(IY+d)

SETX    b,d             SET     b,(IX+d)        SET     b,d(IX)
    Set bit b in memory at m(IX+d)

SETY    b,d             SET     b,(IY+d)        SET     b,d(IY)
    Set bit b in memory at m(IY+d)

RESX    b,d             RES     b,(IX+d)        RES     b,d(IX)
    Reset bit b in memory at m(IX+d)

RESY    b,d             RES     b,(IY+d)        RES     b,d(IY)
    Reset bit b in memory at m(IY+d)

JR      addr            JR      addr-$          JMPR    addr
    Jump relative unconditional

JRC     addr            JR      C,addr-$        JRC     addr
    Jump relative if Carry indicator true

JRNC    addr            JR      NC,addr-$       JRNC    addr
    Jump relative if carry indicator false

JRZ     addr            JR      Z,addr-$        JRC     addr
    Jump relative if Zero indicator true

JRNZ    addr            JR      NZ,addr-$       JRNZ    addr
    Jump relative if Zero indicator false

DJNZ    addr            DJNZ    addr-$          DJNZ    addr
    Decrement B, jump relative if non-zero

PCIX                    JMP     (IX)            PCIX
    Jump to address in IX ie, Load PC from IX

PCIY                    JMP     (IY)            PCIY
    Jump to address in IY

RETI                    RETI                    RETI
    Return from interrupt

RETN                    RETN                    RETN
    Return from non-maskable interrupt

INP     r               IN      r,(C)           INP     r
    Input from port C to register r

OUTP    r               OUT     (C),r           OUTP    r
    Output from register r to port (C)
```

zu Abb. 1.1.2-6

```
INI                     INI                     INI
    Input from port (C) to m(HL), increment HL, decrement b

INIR                    INIR                    INIR
    Input from port (C) to m(HL), increment HL, decrement B, repeat if B <> 0

OUTI                    OTI                     OUTI
    Output from m(HL) to port (C), increment HL, decrement B

OUTIR                   OTIR                    OUTIR
    Repeat OUTI until B = 0

IND                     IND                     IND
    Input from port (C) to m(HL), decrement HL & B

INDR                    INDR                    INDR
    Repeat IND until B = 0

OUTD                    OTD                     OUTD
    Output from m(HL) to port (C), decrement HL & B

OUTDR                   OTDR                    OUTDR
    Repeat OUTD until B = 0

RLCR    r               RLC     r               RLCR    r
    Rotate left circular register

RLCX    d               RLC     (IX+d)          RLCR    d(IX)
    Rotate left circular indexed memory

RLCY    d               RLC     (IY+d)          RLCR    d(IY)
    Rotate left circular indexed memory

RALR    r               RL      r               RALR    r
    Rotate left arithmetic register

RALX    d               RL      (IX+d)          RALR    d(IX)
    Rotate left arithmetic indexed memory

RALY    d               RL      (IY+d)          RALR    d(IY)
    Rotate left arithmetic indexed memory

RRCR    r               RRC     r               RRCR    r
    Rotate right circular register

RRCX    d               RRC     (IX+d)          RRCR    d(IX)
    Rotate right circular indexed

RRCY    d               RRC     (IY+d)          RRCR    d(IY)
    Rotate right circular indexed
```

zu Abb. 1.1.2-6

```
RARR    r               RR      r               RARR    r
    Rotate right arithmetic register

RARX    d               RR      (IX+d)          RARR    d(IX)
    Rotate right arithmetic indexed memory

RARY    d               RR      (IY+d)          RARR    d(IY)
    Rotate right arithmetic indexed memory

SLAR    r               SLA     r               SLAR    r
    Shift left register

SLAX    d               SLA     (IX+d)          SLAR    d(IX)
    Shift left indexed memory

SLAY    d               SLA     (IY+d)          SLAR    d(IY)
    Shift left indexed memory

SRAR    r               SRA     r               SRAR    r
    Shift right arithmetic register

SRAX    d               SRA     (IX+d)          SRAR    d(IX)
    Shift right arithmetic indexed memory

SRAY    d               SRA     (IY+d)          SRAR    d(IY)
    Shift right arithmetic indexed memory

SRLR    r               SRL     r               SRLR    r
    Shift right logical register

SRLX    d               SRL     (IX+d)          SRLR    d(IX)
    Shift right logical indexed memory

SRLY    d               SRL     (IY+d)          SRLR    d(IY)
    Shift right logical indexed memory

RLD                     RLD                     RLD
    Rotate left digit

RRD                     RRD                     RRD
    Rotate right digit
```

zu Abb. 1.1.2-6

1.2 Mikrocomputersystem mit dem Z80

In diesem Abschnitt werden Hinweise und Schaltungen gegeben, die den Aufbau eines größeren Mikrocomputers erlauben. Dem Leser ist es danach möglich, ein, seinen Vorstellungen angepaßtes System, zusammenzustellen.

1.2.1 Speichererweiterung

RAM:

Abb. 1.2.1-1 zeigt das Schaltbild einer einfachen Speichererweiterung für 4K Bytes.

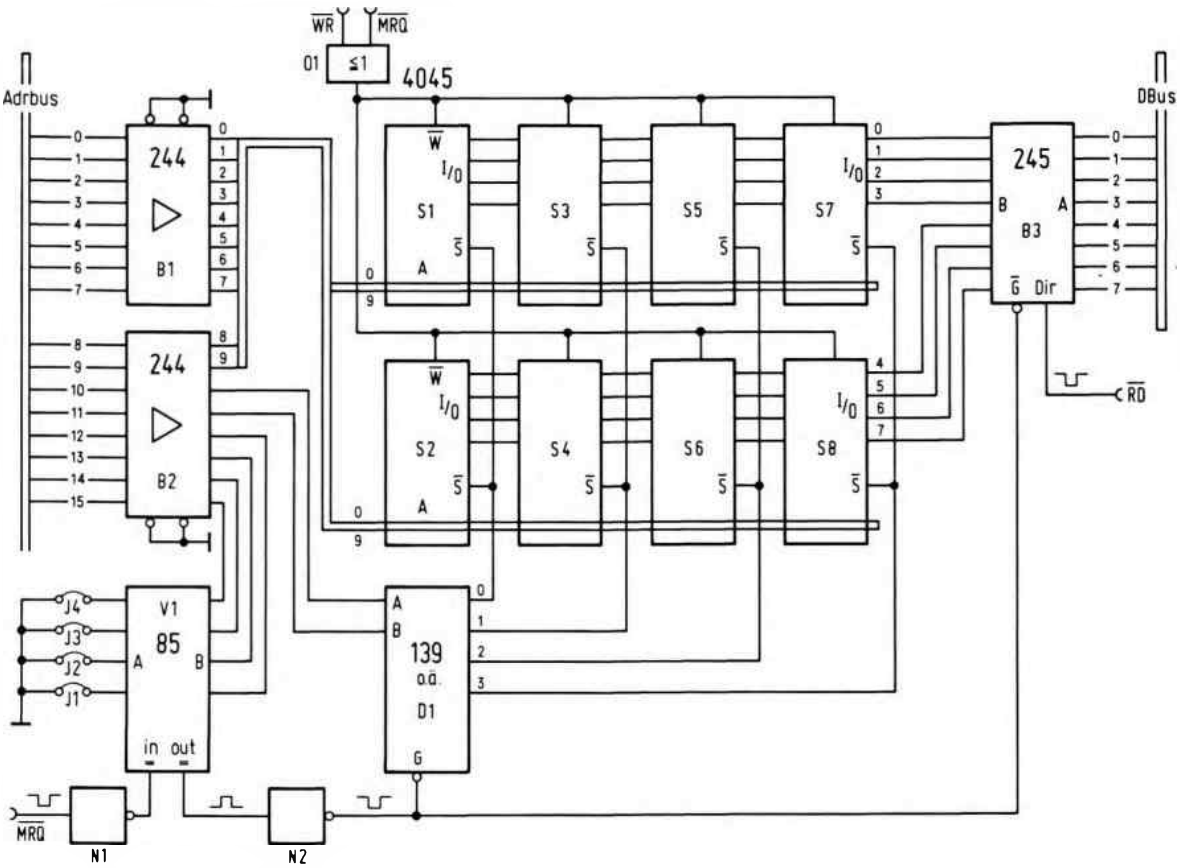


Abb. 1.2.1-1 Einfache Speichererweiterung für den Z80

Der Adreßbereich kann mit den vier „jumpers" J1 bis J4 eingestellt werden, an deren Stelle auch ein DIL-Schalter verwendet werden kann. Als Speicher wurde das IC 4045 von Texas Instrument verwendet, das eine Organisation von 4 x 1024 Bytes besitzt. Das IC arbeitet vollkommen statisch und eine Refreshschaltung kann daher entfallen. Durch die hohe Integrationsdichte ist es möglich, mit nur acht Speicher-ICs einen Speicher von 4K Bytes aufzubauen. Der Adreßbus der CPU wird mit zwei integrierten Schaltungen des Typs 74 LS 244 gepuffert, um ihn nur mit einer TTL-Last (Low power) zu belasten. Falls insgesamt nur 4K Bytes an die CPU angeschlossen werden sollen, können die beiden Treiber B1 und B2 natürlich entfallen.

Die Datenleitungen des Speichers sind mit einem bidirektionalen Bustreiber des Typs 74 LS 245 gepuffert.

Das Speicher-IC besitzt neben seinen vier IO-Leitungen und zehn Adreßleitungen noch zwei Steuereingänge.

Der Eingang $\overline{W}$ bestimmt, ob in den Speicher eingeschrieben werden soll, oder ein Lesezugriff erfolgt.

Bei einem Lesevorgang wird der Eingang auf den Pegel High gelegt.

Die Erzeugung des $\overline{W}$-Signals geschieht mit einer ODER-Verknüpfung 01. Dazu werden die Signale $\overline{MRQ}$ und $\overline{WR}$ miteinander verknüpft. Am Ausgang des ODER-Gatters erscheint nur dann ein Low-Pegel, wenn $\overline{MRQ}$ und $\overline{WR}$ auf Low liegen und damit nur bei einem Speicher-Schreibvorgang.

Der Eingang $\overline{S}$ der Speicher-ICs bestimmt, wann sie ausgewählt (select) werden. Da ein Speicher nur vier Datenleitungen besitzt, beim Z80 aber acht Bit gleichzeitig benötigt werden, sind immer zwei Speicher-ICs zur gleichen Zeit selektiert und daher die $\overline{S}$-Eingänge paarweise verbunden. Es ergeben sich vier Speichergruppen. Die Auswahl dieser Speichergruppen erfolgt mit dem Dekodierer D1 (74 LS 139). Dazu erhält D1 zwei Adressen, A11 und A12, sowie ein enables Signal, das angibt, wann die Speicherkarte adressiert ist.

Dazu ist der $\overline{G}$-Eingang von D1 mit dem Ausgang des Inverters N2 verbunden. Der Vergleicher V1 hat die Aufgabe, die an den Jumpern J1 bis J4 eingestellte Adresse mit der aktuellen Adresse der CPU zu vergleichen, aber nur bei einem Speicherzugriff, und daher ist der „="-Eingang des Vergleichers mit MRQ verbunden (Negation von $\overline{MRQ}$). Der Ausgang von N2 führt auch an den Bustreiber B3, so daß dieser immer dann aktiviert wird, wenn die Karte angesprochen wird. Der Eingang DIR wird von dem Signal $\overline{RD}$ gesteuert, so daß die Datenleitungen immer dann von der Karte zum Rechner geschaltet sind, wenn ein Lesevorgang stattfindet.

Prinzipiell hätte die Steuerung des DIR-Signals auch mit dem Signal $\overline{WR}$ erfolgen können, wenn die Leitungen A mit B vertauscht werden, doch ist diese Betriebsweise nicht empfehlenswert, da bei einem Schreibzugriff die Daten erst mit dem WR-Signal zu den Speichern geschaltet werden und Probleme mit Impulsflanken auftreten, ganz abzusehen davon, daß kurzzeitig zwei Bustreiber (B3 und CPU-Datentreiber) gegeneinander arbeiten. Bei der Steuerung mit RD treten keine Probleme auf, da die kritischen Stellen gerade umgekehrt liegen und die CPU bzw. die Speicher für diese Betriebsweise ausgelegt sind.

Die Kapazität pro Karte kann bis zu 16K Bytes erweitert werden, wenn anstatt des ICs D1 ein IC des Typs 74154 o.ä. verwendet wird. Dann erhält aber das IC V1 nur die oberen beiden Adressen A14 und A15.

ROM:

Abb. 1.2.1-2 zeigt eine Speichererweiterung mit EPROMs, die zum Beispiel zur Aufnahme eines Monitorprogramms dienen können. Die Schaltung ist für 4K Bytes EPROM ausgelegt, kann aber wie im Falle der RAM-Schaltung leicht auf 16K Bytes erweitert werden.

Im Prinzip arbeitet diese Schaltung genauso wie die RAM-Erweiterung, nur daß an Stelle des ICs 74 LS 245 als Treiber der einfache Bustreiber 74 LS 244 verwendet werden kann und das Gatter 01 der RAM-Schaltung entfällt. Mit den Jumpers J1 bis J4 läßt sich der Adreßbereich einstellen, in dem der 4K Block zu liegen kommt.

An Stelle des 2708 können auch neuere EPROMs verwendet werden, wie zum Beispiel der 2732, der eine Kapazität von 4K Bytes besitzt, bei gleicher Packungsdichte. Die Adreßdekodierung muß dann wie folgt geändert werden: Zwei zusätzliche Adressen werden vom Ausgang A10 und A11 des Treibers B2 verwendet, die Leitungen A und B

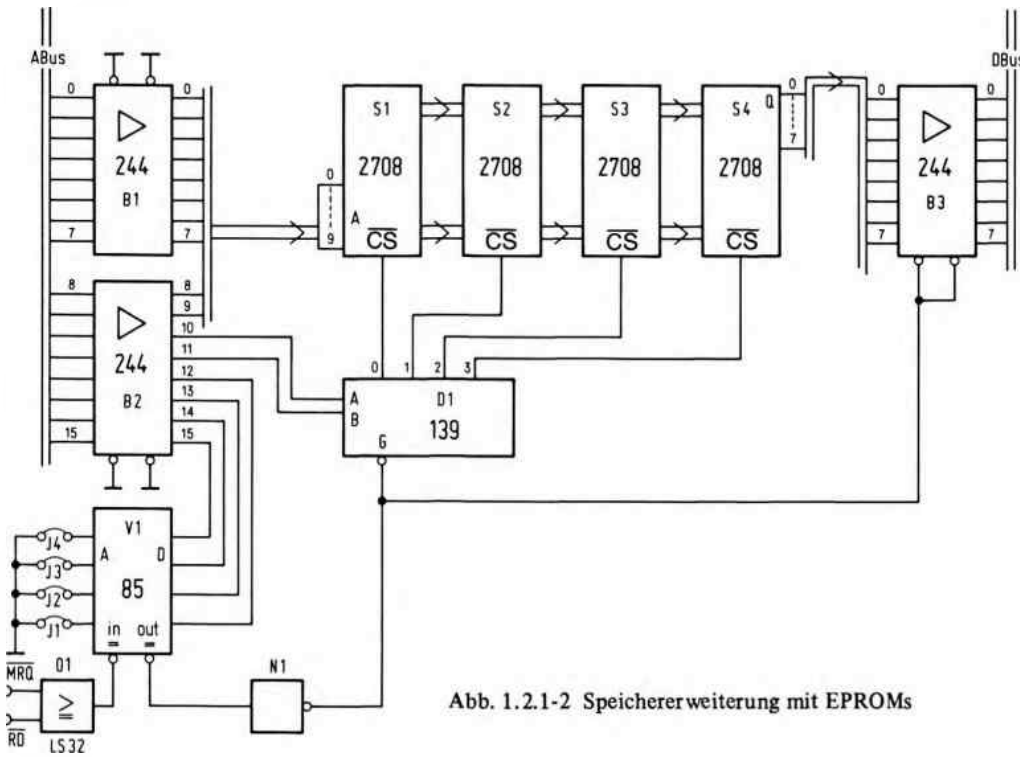


Abb. 1.2.1-2 Speichererweiterung mit EPROMs

des Dekoders D1 werden dann mit A12 und A13 verbunden. Die entsprechenden Eingänge des Vergleichers V1, an denen vorher die Adressen A12 und A13 angeschlossen waren, werden nach Masse geschaltet, wie auch die Eingänge auf der A-Seite des Vergleichers, die mit J1 und J2 verbunden waren. Es wird dann ein 16K Block adressiert.

Erweiterung über 64K hinaus:

Bei dem Z80 können maximal 64K Bytes adressiert werden. Manchmal aber ist es wünschenswert einen größeren Speicher adressieren zu können, zum Beispiel wenn mehrere Benutzer den gleichen Rechner verwenden, oder bei einem Graphicterminal mit hoher Auflösung,oder falls ein größerer Programmspeicher (z.B. mit EPROM) ständig zur Vergügung stehen soll.

Die Erweiterung des Speichers ist zum Beispiel mit einer Bankselektschaltung möglich.

Zur Realisierung dieser Schaltung gibt es sehr unterschiedliche Verfahren.

Denkbar wäre, einfach den gesamten 64K Speicherblock mit einem Port abzuschalten und einen weiteren 64K Block zu selektieren. Dies ist aber nicht sehr sinnvoll, da sich das Programm, das die Umschaltung vorgenommen hat, auch in diesem Speicher befinden muß und nach dem Umschaltvorgang an einer definierten Stelle mit der Programmausführung fortgefahren werden muß. Daher kann zum Beispiel der in *Abb. 1.2.1-3* vorge-

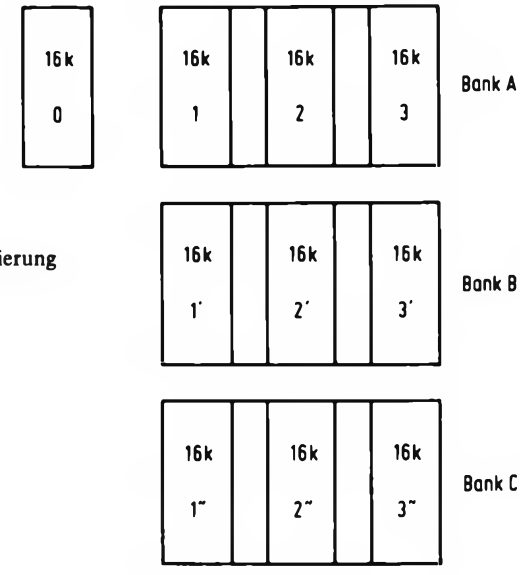


Abb. 1.2.1-3 Einfache Bankselektrealisierung

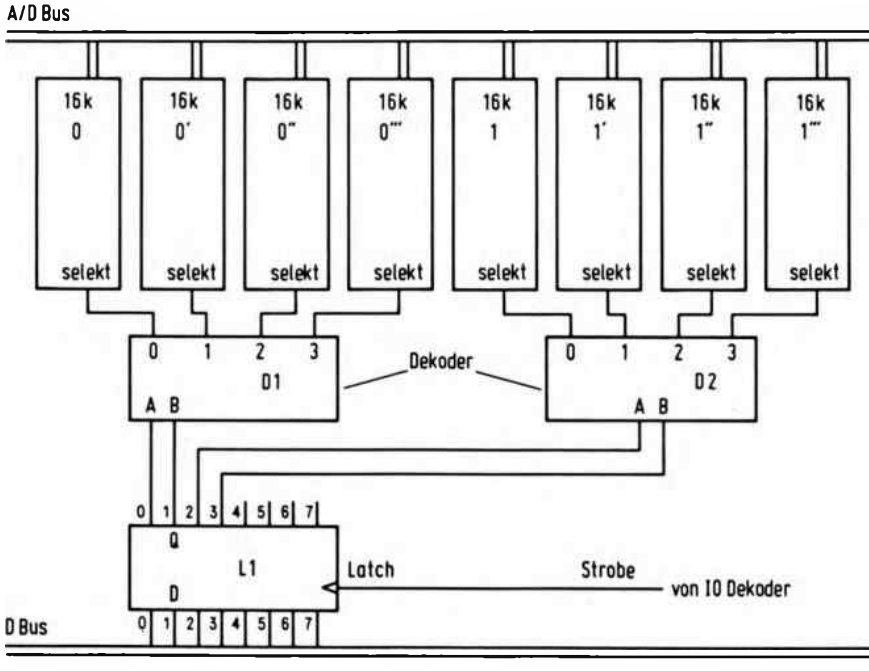


Abb. 1.2.1-4 Universelle Bankselektschaltung

schlagene Weg genommen werden. Hier wird der Speicher in zwei Blöcke aufgeteilt, die getrennt selektierbar sind. Der erste Block ist immer vorhanden und läßt sich nicht wegschalten. Es sind drei Bänke vorhanden, die nun durch einen Ausgabebefehl ausgewählt werden können. Der Umschaltvorgang erfolgt nun immer von Block 0 aus und bewirkt, daß die Blöcke 1, 2 und 3 von einem der drei Bänke A bis C verwendet werden.

Abb. 1.2.1-4 zeigt eine andere, universellere Lösung. Mit dieser Schaltung ist es möglich, von einer beliebigen Stelle aus, einen 16K Block gegen einen anderen auszutauschen. Die Auswahl des gewünschten Blockes erfolgt durch das Einschreiben eines Datenwertes in ein Latch. Die Information des Latches gelangt an mehrere Dekoder, die hier jeweils für einen 16K Speicherbereich eines aus vier Bänken auswählt. Die gesamte Kapazität beträgt hier 256 K Bytes.

1.2.2 „power on jump"

Nach dem Anlegen eines RESET-Impulses an die CPU, wird der Programmzähler auf Null gesetzt, und der Prozessor beginnt dort mit der Ausführung des Programms. Aus diesem Grunde steht bei Entwicklungssystemen fast immer dort das Monitorprogramm. Es gibt aber viele Anwenderprogramme (z.B. BASIC-Interpreter, Assembler), die ebenfalls auf der Adresse 0 anfangen, wie auch zum Beispiel das CPM-Betriebssystem. Es ist folglich notwendig, das Monitorprogramm auf eine andere Adresse zu legen, am besten ganz an den Schluß des Adreßraums, also z.B. von F000H bis FFFFH. Mit einer speziellen Zusatzschaltung muß dann erreicht werden, daß nach einem Resetvorgang die CPU nicht auf Adresse 0 mit der Abarbeitung beginnt, sondern auf der Adresse, bei der das Monitorprogramm steht. Dazu gibt es mehrere Möglichkeiten.

Nach dem RESET kann mit einer Hilfsschaltung ein Sprungbefehl auf dem Datenbus vorgetäuscht werden, mit einer nachfolgenden Adresse, die zum Monitorprogramm zeigt. Der Zugriff zum Speicher wird für diese drei Befehlsbytes gesperrt.

Eine weitere Möglichkeit ist es, unmittelbar nach dem RESET den ROM-Teil mit dem Monitorprogramm auf Adresse 0 zu schalten und den RAM-Bereich auf Adresse 0 zu sperren. In dem Monitorprogramm muß als erster Befehl ein Sprung geschrieben sein, der auf das eigentliche Monitorprogramm innerhalb des ROMs zeigt. Der Sprung, der vom Prozessor aus gesehen auf Adresse 0 liegt, wird nun ausgeführt, und es liegt dann auf dem Adreßbus, die Adresse des Monitorprogramms. Eine weitere Logik erkennt diese Adresse und schaltet den ROM-Bereich auf diesen Adreßbereich zurück, ebenfalls wird der RAM auf Adresse 0 wieder freigegeben. *Abb. 1.2.2-1* zeigt die dazugehörige Schaltung. In die Schaltung gelangen zwei Selektier-Signale, die von einem Adreßdekodierer stammen. Das ROM $\overline{CS}$ selektiert den Monitorbereich, z.B. 4K von Adresse F000H bis FFFFH. RAM $\overline{CS}$ selektiert den Bereich 0 bis FFFH, also ebenfalls 4K Bytes. Im Betriebszustand ist das RS-Flipflop, bestehend aus N1 und N2 so gekippt, daß die beiden Selektsignale ungehindert an die Ausgänge von U1 bzw. 02 geschaltet sind. Nach dem RESET-Impuls kippt das RS-Flipflop in die andere Lage, und am Ausgang von N2 liegt ein High-Pegel, der an den Eingang von 02 führt. Damit erscheint am Ausgang von 02 immer ein High-Pegel, egal welchen Zustand der andere Eingang einnimmt. Der Ausgang von N1, der einen Low-Pegel führt, gelangt an die ODER-Verknüpfung 01, und damit wird der am Eingang dieses Gatters anliegende Pegel an seinen Ausgang durchgeschaltet. Wird nun der Adreßbereich 0 angesprochen, so gelangt ein Low-Pegel an den Eingang von 01 und damit erscheint es auch am Eingang von U1. Durch die

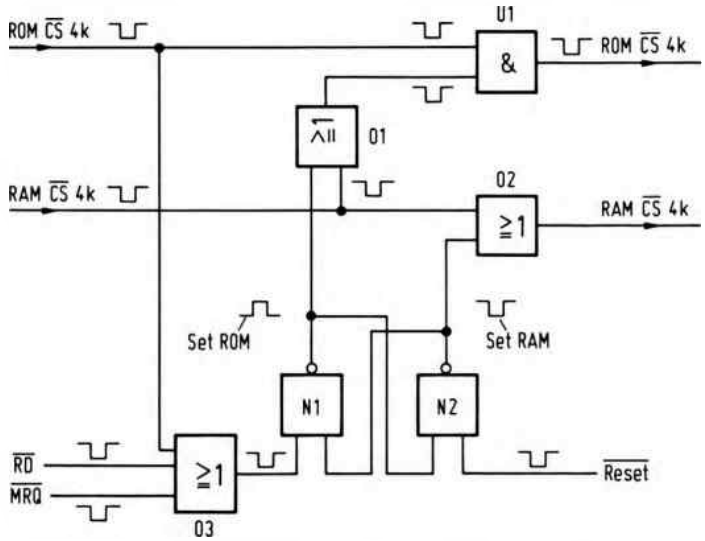


Abb. 1.2.2-1 „Power on jump" mit Vertauscherschaltung

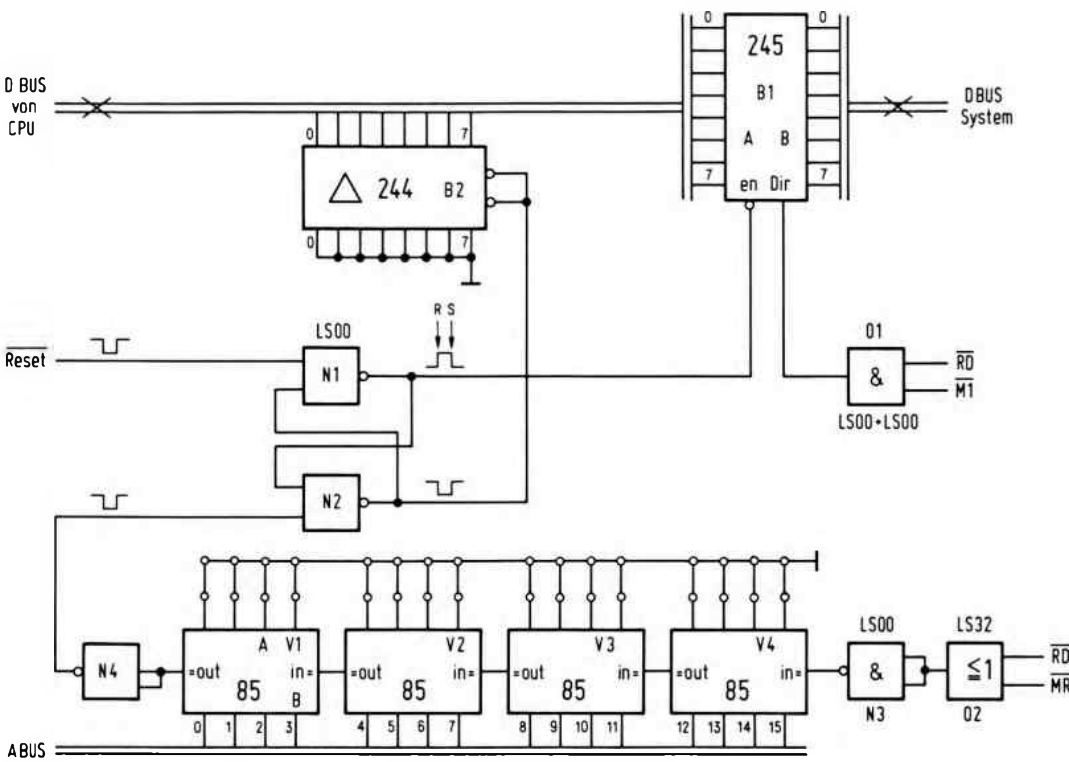


Abb. 1.2.2-2 „Power on jump" mit NOP-Befehlen

UND-Verknüpfung wird dann der Low-Pegel an den Selekt-Eingang der ROM-Baugruppe erreicht, und der Prozessor kann die Befehle auslesen. Erfolgt nun ein Zugriff auf den Monitorbereich, so wird ROM $\overline{CS}$ angesprochen und ein Low-Pegel gelangt an die ODER-Verknüpfung 03. An die anderen beiden Eingänge von 03 gelangen die Signale $\overline{RD}$ und $\overline{MRQ}$. Am Ausgang von 03 erscheint nur dann ein Low-Pegel, wenn ein Speicherzugriff (lesend) auf den Monitorbereich erfolgt und das RS-Flipflop kippt in die Ruhelage zurück. *Abb. 1.2.2-2* zeigt eine weitere Lösung für das „power on jump"-Problem. Mit dieser Schaltung ist es möglich, unabhängig von der restlichen Hardware und Software nach einem Resetvorgang auf eine beliebige Stelle im Speicher zu springen. Die Sprungadresse wird an den Vergleicherstufen V1 bis V4 mit Hilfe von „jumpers" oder DIL-Schaltern eingestellt. Nach Erscheinen eines RESET-Signals wird das RS-Flipflop rückgesetzt, und der Ausgang von N1 nimmt einen High-Pegel an. Damit wird der Bustreiber B1 gesperrt und die A- und B-Seite nimmt einen hochohmigen Zustand an. Gleichzeitig erscheint am Freigabeeingang von B2 ein Low-Pegel, und der Bustreiber schaltet die Low-Pegel seiner Eingänge auf den Datenbus durch. Sind aber alle Leitungen des Datenbus auf Low, so sieht die CPU einen NOP-Befehl bei der Ausführung.

Bei der Ausführung dieses Befehls wird der Adreßzähler um eins erhöht. Dies geschieht nun solange, bis die Vergleicherstufen V1 bis V2 eine Adresse auf dem Adreßbus vorfinden, die mit der eingestellten Adresse übereinstimmt und außerdem ein Speicherlesezugriff stattfindet. Das RS-Flipflop wird dann gesetzt und der Bustreiber B2 wieder gesperrt. B1 wird freigegeben und die CPU führt das an dieser Adresse stehende Programm aus. Die gezeichnete Schaltung wird zwischen der CPU-Karte und der restlichen Speicherschaltung angebracht.

1.2.3 Parallelports

Jeder Mikrocomputer muß irgendwie mit der Außenwelt in Verbindung treten. Am einfachsten geschieht dies über Parallelports. Eine einfache Möglichkeit wurde schon in Abb. 1.1.1-22 vorgestellt. Hier wurden als Parallelports einfache TTL-Bausteine verwendet. Wenn die Anzahl der IO-Leitungen und deren Bedeutung vonvornherein bestimmt ist, so kann durchaus so verfahren werden. Soll aber eine Universalität erreicht werden, da sich im Verlauf einer Systementwicklung Änderungen ergeben können, so kommen nur spezielle LSI-Schaltungen in Frage, deren IO-Eigenschaften vom Prozessor programmiert werden können. Es sind derzeit eine Vielfalt von IO-Bausteinen verfügbar für die unterschiedlichsten Prozessoren.

Hier soll nun ein typischer Parallelport-Baustein besprochen werden.

8255

Es handelt sich bei dem Baustein 8255 der Firma INTEL um einen universalen Parallelport mit 24 IO-Leitungen. Dieses IC wurde ursprünglich für den 8080 ausgelegt, ist aber auch für andere CPUs verwendbar, wie zum Beispiel für den Z80.

Der 8255 besitzt drei Ports mit jeweils 8-Bit-Leitungen. Diese Ports sind mit A, B und C bezeichnet.

Die beiden Ports A und B sind praktisch gleichwertig, nur der Port C unterscheidet sich von ihnen wesentlich, da er noch einmal in zwei 4-Bit Hälften gespalten ist, deren IO-Eigenschaft getrennt programmiert werden kann. Für die Kanäle A und die oberen

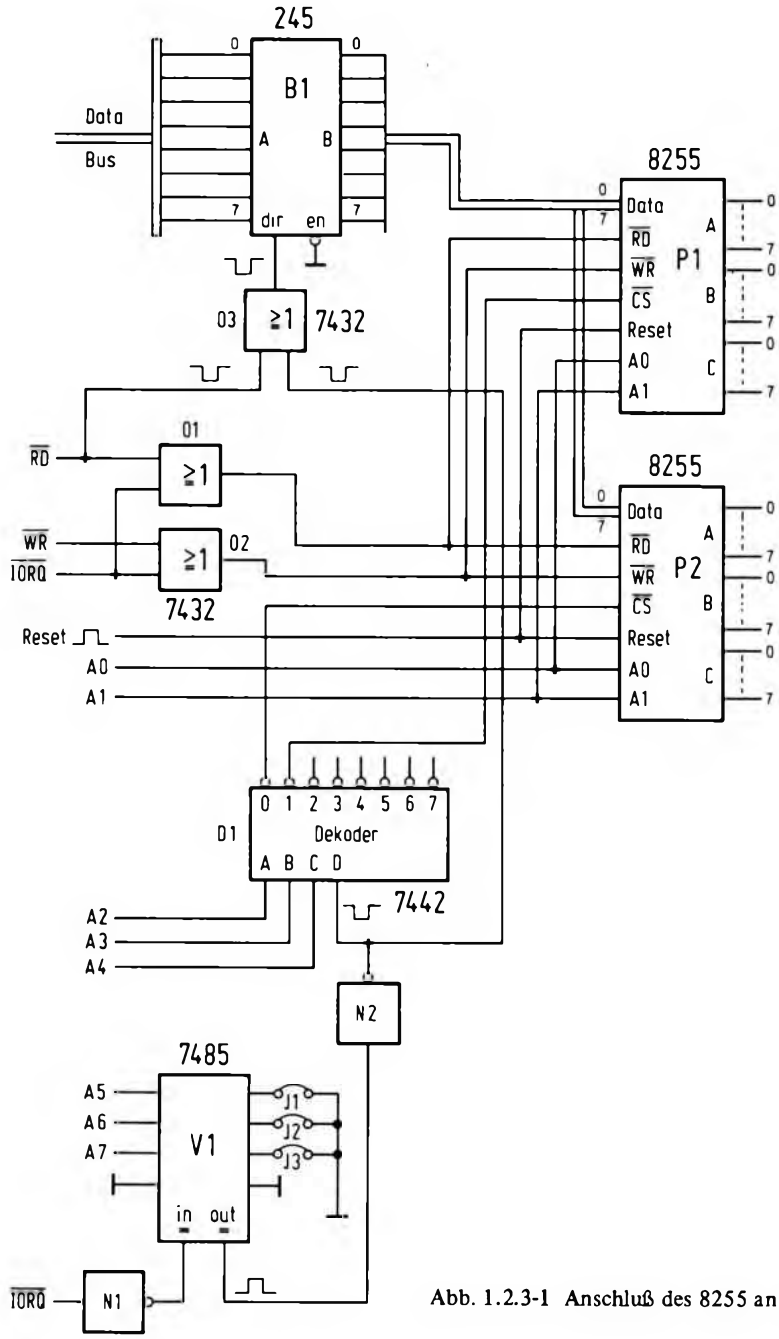


Abb. 1.2.3-1 Anschluß des 8255 an den Z80

4 Bit von C sind drei verschiedene Betriebsarten einstellbar, für die Kanäle B und die unteren 4 Bit von C sind zwei Betriebsarten programmierbar. In der Betriebsart 0 kann die Datenrichtung von jedem dieser vier Datenkanäle unabhängig voneinander eingestellt werden. In der Betriebsart 1 wird ein Teil von Port C für Handshakesignale verwendet und die Betriebsart 2 schließlich erlaubt es über den Port B einen bidirektionalen Datentransfer abzuwickeln.

Auch hier werden Leitungen des Port C für Handshakesignale benötigt. Die genauen Programmierdaten können in dem Benutzerhandbuch für den 8255 der Firma INTEL (16) nachgelesen werden.

Abb. 1.2.3-1 zeigt eine einfache Schaltung zum Anschluß des 8255 an den Z80. Es sind hier zwei Bausteine eingezeichnet, die Dekodierschaltung läßt aber insgesamt acht Bausteine zu. Der 8255 besitzt vier interne Register, die über eine Adresse ausgewählt werden. Dazu gelangen die Leitungen A0 und A1 des Bausteins direkt an den CPU-Adreßbus. Mit den Eingängen $\overline{RD}$ und $\overline{WR}$ wird die Zugriffsart bestimmt, und die Signale werden auf die schon bekannte Weise mit Hilfe zweier ODER-Verknüpfungen von den CPU-Signalen $\overline{RD}$, $\overline{WR}$ und $\overline{IORQ}$ gewonnen. Der RESET-Eingang stellt den Grundzustand des 8255 her, indem die Ports A bis C in der Betriebsart 0 als Eingänge geschaltet werden. Mit dem Eingang $\overline{CS}$ wird der Baustein selektiert. Dazu ist dieser Eingang mit dem Ausgang des Dekoders D1 verbunden. Der Dekoder erhält die Adressen A2 bis A3 und kann damit acht 8255-Bausteine auswählen. Die Freigabe des Dekoders erfolgt über den Eingang D, der dazu über einen Inverter mit dem Ausgang des Vergleichers V1 verbunden ist. Wenn die mit den „jumpers" J1 bis J3 eingestellte Adresse mit A5 bis A7 übereinstimmt, und ein IO-Zugriff vorliegt, so erscheint an dessen Ausgang ein High-Pegel.

Das Selekt-Signal gelangt über den Inverter N2 außerdem an 03. Mit dem Ausgang dieser ODER-Verknüpfung wird die Richtung des Bustreibers B1 gesteuert. Bei einem Lesezugriff wird die Richtung von den Bausteinen zum Prozessorbus geschaltet, ähnlich wie bei der RAM-Erweiterungsschaltung.

1.2.4 Serienports

Sollen Daten über eine größere Entfernung übertragen werden, so ist es angenehm, wenn dies über wenige Leitungen möglich ist. Für den normalen Datenaustausch bei Mikrocomputern werden aber normalerweise 8 Bit verwendet, daher ist es nötig, diese acht Bit in ein serielles Format umzuwandeln. Wie dieses serielle Format aussieht, wurde schon in dem „Mikrocomputersysteme" (1) beschrieben, daher soll hier nur die Schaltung besprochen werden.

Für Mikroprozessoren gibt es eine Vielfalt an hochintegrierten Schaltungen, die eine Serialisierung vornehmen können. Hier wird der recht preiswerte Serienport 8251 besprochen, der sowohl asynchrone als auch synchrone Übertragung handhaben kann.

Abb. 1.2.4-1 zeigt die Schaltung. Als Bussteuerung kann der gleiche Schaltungsteil wie in Abb. 1.2.3-1 verwendet werden. Die Datenleitungen D0 bis D7 werden an den internen Datenbus angeschlossen, $\overline{RD}$, $\overline{WR}$ und RESET werden genauso wie bei dem Parallelport 8255 angeschlossen.

Da der Baustein im Gegensatz zum 8255 nur zwei interne Register besitzt, die mit $C/\overline{D}$ ausgewählt werden, bleibt die Adreßleitung A1 frei. Soll der Adreßraum voll ausgenützt werden, so muß die Dekodierschaltung geändert werden. Der Dekoder D1 in

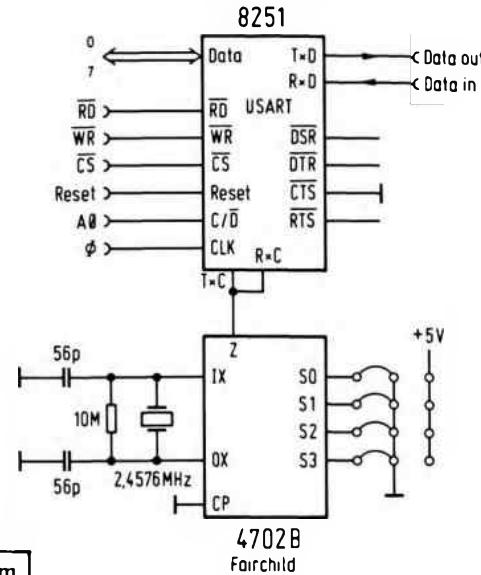


Abb. 1.2.4-1 Anschluß des 8251 an den Z80

Tabelle 1.2.4-1

S3	S2	S1	S0	Baudrate
L	L	L	L	Multiplexed Input Im
L	L	L	H	Multiplexed Input Im
L	L	H	L	50 Baud
L	L	H	H	75 Baud
L	H	L	L	134.5 Baud
L	H	L	H	200 Baud
L	H	H	L	600 Baud
L	H	H	H	2400 Baud
H	L	L	L	9600 Baud
H	L	L	H	4800 Baud
H	L	H	L	1800 Baud
H	L	H	H	1200 Baud
H	H	L	L	2400 Baud
H	H	L	H	300 Baud
H	H	H	L	150 Baud
H	H	H	H	110 Baud

Abb. 1.2.3-1 erhält dann die Adressen A1 bis A3 und der Vergleicher A4 bis A7.

Der $\overline{CS}$-Eingang des Bausteins 8251 wird an den gewünschten Ausgang des Dekoders D1 geschaltet, je nach Adreßbereich. Der 8251 erhält ferner noch ein Taktsignal CLK, das am einfachsten aus dem CPU-Takt Φ gewonnen wird.

Der USART 8251 ist hier für den asynchronen Mode beschaltet, wie er bei normalen Terminals üblich ist (u.a. auch Teletype). Er erhält einen Bittakt an die Eingänge TxC und RxC von einem Baudrate Generator des Typs 4207B.

Dieser erzeugt die 16fache Frequenz von den gebräuchlichsten Baudraten. Dazu wird er von einer Quarzfrequenz gesteuert, wobei die Schaltung zur Erzeugung der Schwingung mit integriert ist. Die Quarzfrequenz beträgt 2,4576 MHz.

Tabelle 1.2.4-1 zeigt die möglichen Baudraten, wobei die ersten beiden Einstellungen für eine externe Frequenzeinspeisung gedacht sind.

V24/20 mA Schnittstellen:

Über Leitungen können mit normalen TTL-Pegeln (0V und 5V) keine allzu großen Entfernungen störungsfrei überbrückt werden, dazu ist es nötig, mit größeren Spannungshüben zu arbeiten, oder einen eingeprägten Strom zu verwenden. Eine Schnittstelle mit genormten Spannungspegeln ist die V24-Schnittstelle. Es wird mit einem Pegel von - 12 V und + 12 V gearbeitet. Als Stromschnittstelle sei zum Beispiel die 20-mA-Schnittstelle genannt.

Mit beiden Schnittstellen kann eine Entfernung von mindestens 100 Meter überbrückt werden, wobei die Stromschnittstelle noch günstiger ist. Mit TTL–ICs (z.B. 7400) gibt es schon bei ein paar Meter Leitungslänge erhebliche Schwierigkeiten. Die Entfernungsangabe ist natürlich abhängig von der Übertragungsgeschwindigkeit, hier wurde von 9600 Baud ausgegangen. *Abb. 1.2.4-2* zeigt eine Schaltung, die sowohl V-24-Signale, als auch die 20-mA-Stromschnittstelle erzeugen kann. Die Betriebsart ist mit insgesamt vier Drahtbrücken (jumpers) einstellbar. Die obere Bildhälfte zeigt den Sendeteil.

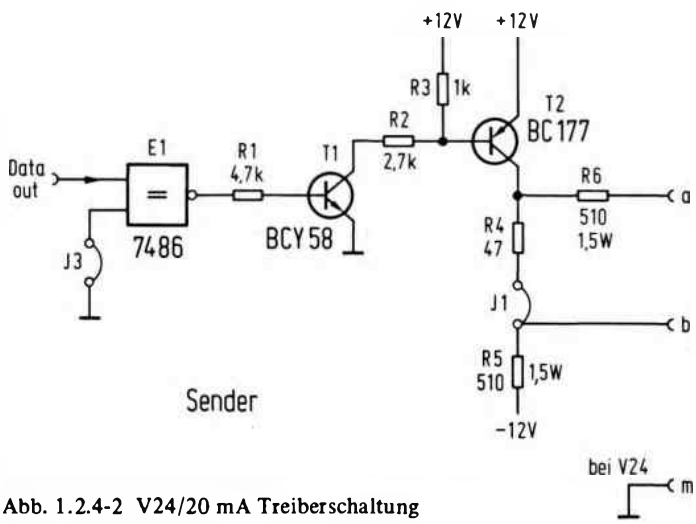
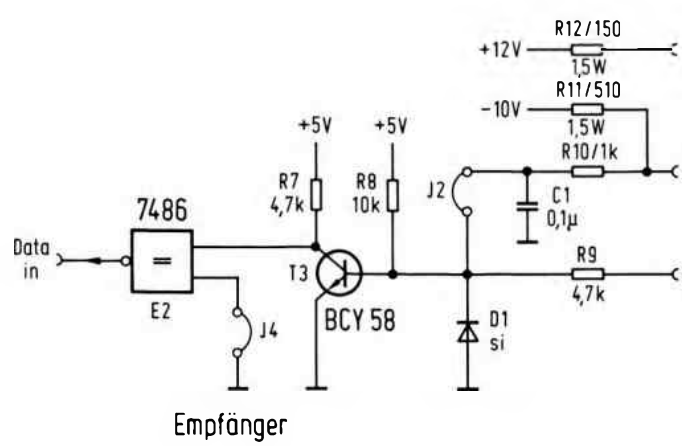


Abb. 1.2.4-2 V24/20 mA Treiberschaltung



Das Signal „data out" gelangt von einem USART (z.B. Abb. 1.2.4-1) an ein Exclusiv-ODER-Gatter E1. Mit J3 kann die Ausgangspolarität eingestellt werden. Bei V24 gilt der – 12 V Pegel als Ruhelage, bei der 20-mA-Schnittstelle muß in der Ruhelage ein Strom fließen. Das Datensignal gelangt vom Ausgang des Gatters E1 an einen Treibertransistor T1. Von dort aus wird das Signal an T2 übertragen. Dieser Transistor ist der eigentliche Schnittstellentreiber. Mit J1 wird bestimmt, ob die Schaltung für den V-24-Betrieb oder für 20 mA verwendet werden soll. Ist J1 eingesetzt, so erscheint am Ausgang b das V-24-Signal. Ist J1 offen, so wird an der Klemme a +TX übertragen und die Klemme b dient der Rückführung (–TX). Der Strom errechnet sich zu 24V/(R6+R5)= 23 mA, ist etwas größer als 20 mA, aber nur ohne Berücksichtigung des Leitungswiderstandes. Am anderen Ende der Leitung befindet sich meist eine Optokopplerstufe, die das Signal auf TTL-Pegel transformiert. Im unteren Bildteil ist der Empfänger abgebildet. Das Datensignal am Ausgang von T3 gelangt über ein Exclusiv-ODER-Gatter an den USART-Eingang. Mit J4 kann die Polarität des Ruhepegels eingestellt werden, der bei dem USART 8251 auf High liegen muß.

Im 20-mA-Betrieb ist die Brücke J2 eingefügt, und die Daten werden über die Leitungen c und d übertragen. Dabei stellt die Leitung c den Anschluß +RX dar und wird daher mit –TX des Senders verbunden, und d ist –RX. Zu beachten ist, daß es sich um eine aktive Schnittstelle handelt, die nicht mit einer anderen aktiven verbunden werden darf. Auch ist es nicht möglich, die Schnittstelle im Kreis zu führen, da hier Sender und Empfänger nicht galvanisch entkoppelt sind.

Im Ruhezustand fließen etwa 20 mA über die Leitung von c nach d. R12 und R11 bilden damit einen Spannungsteiler. Es liegt im Ruhezustand an Punkt d, bezogen auf Masse ein Pegel von etwa 7V an. Ist die Schleife offen, so liefen dort –10 V an. T3 schaltet also, wenn der Ruhezustand vorliegt.

Im V-24-Betrieb ist J2 offen, und der Eingang e wird für die V-24-Signale verwendet. Die Diode D1 dient der Begrenzung für negative Spannungspegel, die an der Basis von T3 entstehen könnten.

Ein deutlicher Unterschied zwischen V-24- und 20-mA-Schnittstellen besteht auch in der Anzahl der Leitungen. Für V-24 genügen drei (b, e, m) für die 20-mA-Schnittstelle werden aber vier (a, b, c, d) benötigt, die aber paarweise verdrillt werden können und damit auch einen sichereren Betrieb ermöglichen.

1.3 Peripherieschaltungen

1.3.1 Displays-Datensichtgeräte

1.3.1.1 CRT-Controler und ihre Anwendung

Seit geraumer Zeit gibt es auf dem Markt eine Vielzahl an CRT-Controler-Bausteinen, die es ermöglichen, mit geringem Aufwand Datensichtgeräte zu konstruieren.

Diese Controler werden meist mit einem Mikroprozessor verbunden, der die Steuerung der Datenwiedergabe auf dem Bildschirm übernimmt. Der Controler hat die Aufgabe, die Synchronisationssignale für den Monitor (oder TV-Gerät) zu erzeugen, sowie die Adressen für den Bildwiederholspeicher. Ferner wird über eine Vergleicherlogik die Position des Cursors bestimmt.

Die Anzahl der Zeilen, Bildschirmweite, Synchronisationsart und Lage der Impulse können vom Mikroprozessor programmiert werden.

Hier werden zwei repräsentative Bausteine besprochen, die alle diese Eigenschaften vorweisen.

1. CRT-Video-Timer-Controler VTAC 5027:

Der Baustein erlaubt es, bis zu 132 Zeichen pro Zeile zu programmieren, sowie bis zu 16 Punkte pro Zeile in vertikaler Richtung. Dabei ist der kleinste einstellbare Wert 20 Zeichen pro Zeile und 1 Punkt pro Zeichen. Das IC ist voll TTL kompatibel und kann an den meisten Prozessorsystemen direkt betrieben werden.

Abb. 1.3.1.1-1 zeigt die PIN-Belegung des CRT-Controlers. Er besitzt vier Adreßeingänge zur Adressierung der internen Arbeitsregister.

Tabelle 1.3.1.1-1 zeigt die Bedeutung der einzelnen Register. Die Adressen 0 bis 3 bestimmen eines der sechs Control-Register zur Steuerung des Bildformats.

Tabelle 1.3.1.1-2 zeigt das dazugehörige Format.

Eine Besonderheit ist der Selbstlademechanismus, der es erlaubt, auch ohne Prozessor auszukommen. Wird die Adresse FH angewählt, also alle Eingänge auf High-Pegel, so werden über R0 bis R3 Adressen ausgegeben, die an einen PROM geführt werden können, dessen Ausgang mit dem Datenbus verbunden ist. Nach einem LOW-Signal auf der $\overline{DS}$-Leitung wird der Selbstladevorgang eingeleitet.

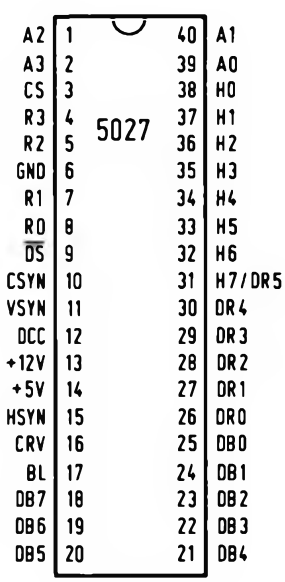


Abb. 1.3.1.1-1 PIN-Belegung des CRT-Controlers 5027

Tabelle 1.3.1.1-1

A3	A2	A1	A0	Befehl	Beschreibung
0	0	0	0	Reg 0	Tabelle 1.3.1.1-2
0	0	0	1	Reg 1	"
0	0	1	0	Reg 2	"
0	0	1	1	Reg 3	"
0	1	0	0	Reg 4	"
0	1	0	1	Reg 5	"
0	1	1	0	Reg 6	"
0	1	1	1	Prozessor Selbst laden	Befehl des Prozessors an CRT in den Selbstlademode zu gehen
1	0	0	0	lesen Cursor Zeilenadresse	
1	0	0	1	lesen Cursor Zeichenadresse	
1	0	1	0	RESET	Die interne Zählerkette wird zurückgesetzt und stoppt bis zum Start Befehl
1	0	1	1	UP SCROLL	die Adresse der ersten angezeigten Datenreihe wird um eins erhöht. Vorher oberste Zeile 0 und unterste 31. Nachher oberste Zeile 1 und unterste 0.
1	1	0	0	schreiben Cursor Zeichenadresse	
1	1	0	1	schreiben Cursor Zeilenadresse	
1	1	1	0	Start der Zählerkette	Startet die Zählerkette nach ungefähr einer Zeile, nach einem RESET oder Prozessor Selbstladevorgang.
1	1	1	1	CRT Selbst laden	Der CRT Controler beginnt den Selbstladeprozeß über einen PROM wenn $\overline{DS}$ auf LOW geht. Die Adressen werden über R0 bis R3 ausgegeben.

Abb. 1.3.1.1-2 zeigt die ausgeführte Schaltung für eine Steuerung mit einer CPU.

Der Datenbus wird, wie schon in den vorherigen Kapiteln, mit einem Bustreiber bidirektional gepuffert. Die Datenrichtung wird durch das Signal am Ausgang von N2 gesteuert. Es nimmt immer dann einen Low-Pegel an, wenn der Baustein adressiert wird (mit V1) und ein IO-Lesezugriff vorliegt.

Der CRT-Controler besitzt keinen R/$\overline{W}$-Eingang, da die Datenrichtung immer eindeutig durch die Registeradresse festgelegt ist. Eigentlich wäre nun eine Schutzschaltung für den Datenbustreiber vorzusehen, die eine falsche Zugriffsart verhindert, doch darauf wurde zugunsten der Schaltungsvereinfachung verzichtet. Der Baustein kann durch eine falsche Zugriffsart außerdem laut Datenblatt nicht zerstört werden.

Tabelle 1.3.1.1-2

Register	Beschreibung	
0	Horizontaler Zeichenzähler gibt die gesamte Zahl der Zeichen pro Zeile an, auch der nicht dargestellten. Von 0 bis 255 einstellbar. Es werden N+1-Zeichen verwendet.	7 0
1	Dies ist ein Register, das mehrere Funktionen erfüllt. Bit 7 bestimmt, ob im Zeilensprungverfahren gearbeitet wird oder nicht. Bei Zeilensprung ist das Bit auf 1 gesetzt. Bit 6 bis 3 bestimmen die H-SYNC-Weite in Zeichenbreiten gerechnet. Bit 2 bis 0 sind für die Sync Verzögerung, mit maximal 8 Zeichenbreiten.	76 32 0
2	Auch dies ist ein Mehrfunktionsregister. Bit 6 bis 3 ist die Anzahl der Zeilen für ein Zeichen von 0 bis 15, wobei N+1-Zeilen verwendet werden. Bit 2 bis 0 bestimmen die Anzahl der dargestellten Zeichen pro Zeile, wobei von 0 bis 7 gilt: 20, 32, 40, 64, 72, 80, 96, 132 Zeichen/Zeile.	6 32 0
3	Mit Bit 7 und 6 werden die Verzögerungs-Zahlen festgelegt, 1 2 zwischen Sync und Blank und eine zwischen Cursor und Adresse, 2 bedeutet 1 zwischen Sync und Blank, keine bei Cursor, und 3 bedeutet 2 zwischen Sync und Blank und 2 zwischen Cursor und Adresse. Bit 5 bis 0 bestimmen die Anzahl der Zeichenzeilen von 0 bis 63, wobei N+1 dargestellt werden.	765 0
4	Dieser Wert bestimmt die Anzahl der Zeilen des Gesamtbildes, wobei für das Zeilensprungverfahren die Formel 2X+512 gilt und für das Nicht-Zeilensprungverfahren 2X+256 gilt.	7 0
5	Start der Daten bei Erreichen der hier angegebenen Zeile nach dem Sync-Impuls.	7 0
6	Mit den Bits 5 bis 0 kann die Adresse der zuletzt dargestellten Datenreihe für Scrollzwecke angegeben werden.	5 0

Die Daten werden mit $\overline{DS}$ in den Controler übernommen, dazu ist dieser Eingang mit $\overline{IORQ}$ verbunden. Über den Eingang CS erfolgt die Freigabe, wenn die richtige IO-Adresse anliegt. Die unteren vier Adreßleitungen A0 bis A3 gelangen über den Bustreiber B2 direkt an den Controler. Es werden 16 Adreßplätze mit dieser Schaltung belegt.

Der Adreßbereich kann mit den vier „jumpers" J1 bis J4 eingestellt werden.

Der CRT-Controler besitzt verschiedene Ausgänge zur Displaysteuerung und zur Steuerung des Bildwiederholspeichers.

Die Ausgänge R0 bis R3 bestimmen die darzustellende Zeile eines Zeichens. Dabei können bis zu maximal 16 Zeilen für ein Zeichen verwendet werden. Dadurch

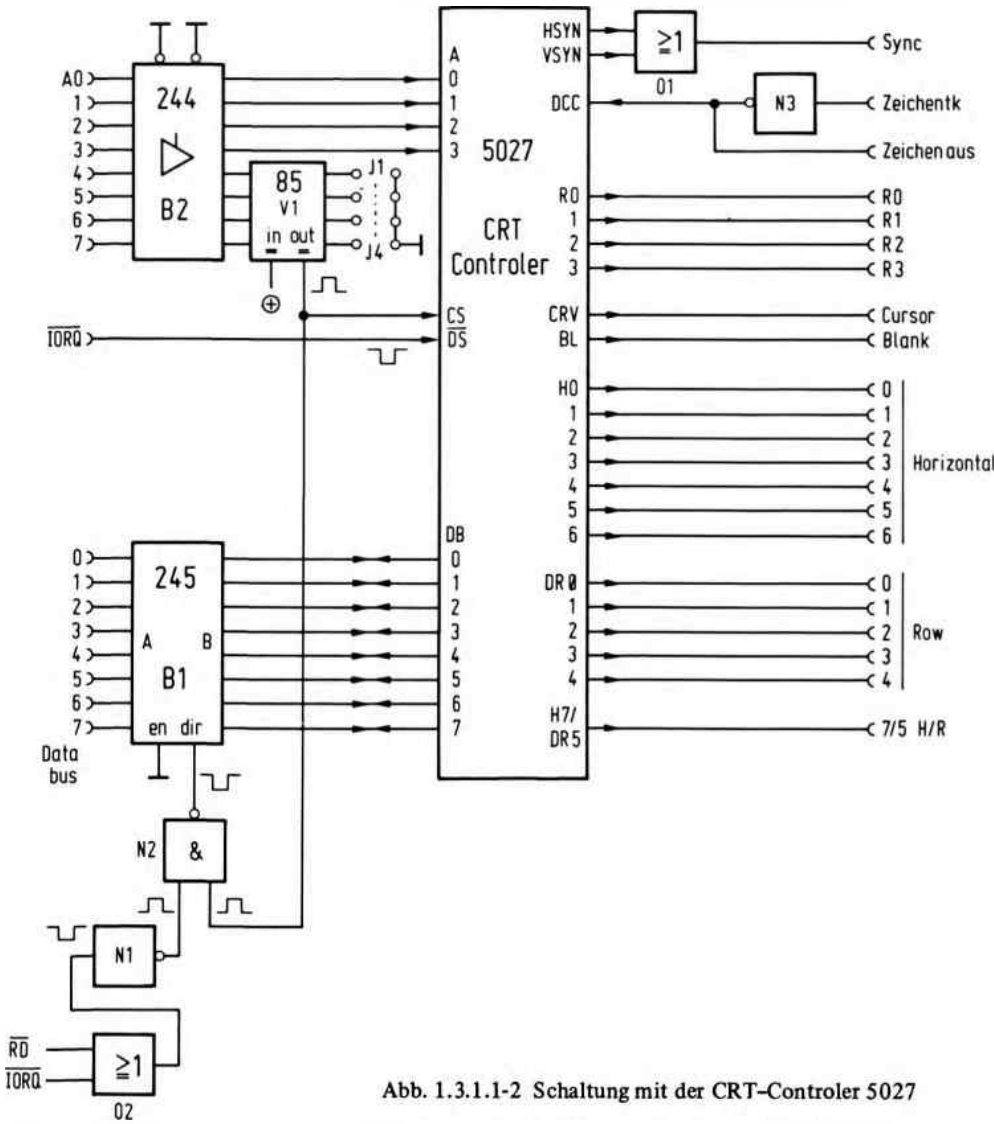


Abb. 1.3.1.1-2 Schaltung mit der CRT-Controler 5027

läßt sich eine sehr hohe Schriftqualität erreichen. Die Anzahl der Punkte pro Zeichen in horizontaler Richtung wird durch eine externe Schaltung bestimmt.

H0 bis H6 dienen der Adressierung von den Zeichen in horizontaler Richtung. H7/DR5 besitzt eine programmierbare Doppelfunktion. Ist das Bit 2 des Registers 2 auf 1 eingestellt, so gilt H7 und die Anzahl der Zeichen in vertikaler Richtung wird eingeschränkt. Ist das Bit 2 von R2 auf 0 programmiert, so gilt die Bezeichnung DR5. Zusammen mit DR0 bis DR4 wird damit die Anzahl der Zeichen in vertikaler Richtung bestimmt.

HSYN und VSYN werden über die ODER-Verknüpfung 01 zu einem gemeinsamen SYNC-Signal gemischt, das direkt einem BAS-Mischer zugeführt werden kann. Der CRT-Controler besitzt auch einen CSYNC (composite sync)-Ausgang, der aber nur dann ein Signal der RS-170-Norm enthält, wenn kein Zeilensprungverfahren verwendet wird. R0 bestimmt im Falle des Zeilensprungverfahrens die geraden, bzw. die ungeraden Zeilen.

Der Ausgang BL dient dem Dunkeltasten des Bildschirmrandes, in dem ja keine Zeichen erscheinen sollen.

DCC ist der Eingang für den Zeichentakt.

Dieser CRT-Controler besitzt auch einen Nachteil, wird eine Zahl von Zeichen pro Zeile eingestellt, die nicht einer Zweierpotenz entspricht, wie zum Beispiel 80 Zeichen pro Zeile, so bleiben die restlichen Speicherzellen zur nächsthöheren Zweierpotenz ungenutzt (hier 128, also 48 Zellen pro Zeile). Es muß daher, falls auf diese Zellen nicht verzichtet wird, eine spezielle Adreßkompressionsschaltung entworfen werden, die diesen Nachteil vermeidet. Diesen Nachteil vermeidet der CRT-Controler 6845.

2. CRT-Controler 6845:

Dieser Baustein besitzt zwei unabhängige Zählerketten. Die eine Zählerkette zählt von einem programmierbaren Anfangswert linear bis zu einer Endadresse, so daß keine Lücken bei der Bildwiederholspeicheradressierung entstehen. Die andere Zählerkette bestimmt den Ablauf der Synchronsignale und arbeitet ähnlich wie beim 5027. Dadurch, daß die Anfangsadresse frei programmierbar ist, kann eine Art Blättern in einem viel größeren Bildwiederholspeicher ermöglicht werden.

Der Baustein kann einen Adreßraum von 16K bedienen. Der Controler benötigt außerdem nur eine 5-V-Versorgung und besitzt einen Lichtgriffeleingang.

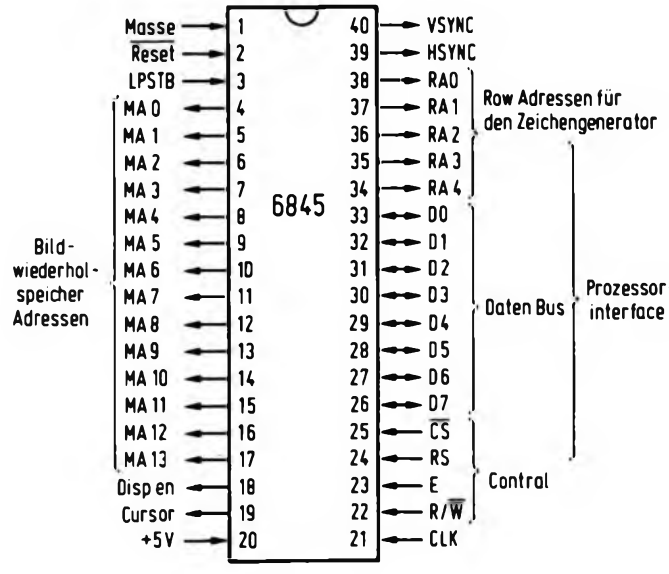


Abb. 1.3.1.1-3 PIN-Belegung des CRT-Controlers 6845

Tabelle 1.3.1.1-3

Adreß-Register	Register Name	Bezeichnung	R/W	Einheit
00000	R0	Horizontal gesamt	W	Zeichen
00001	R1	Horizontal angezeigt	W	Zeichen
00010	R2	HSYNC Position	W	Zeichen
00011	R3	HSNYC Breite	W	Zeichen
00100	R4	Vertikal gesamt	W	Zeichenzeilen
00101	R5	Vertikal Abgleich	W	Bildzeilen
00110	R6	Vertikal angezeigt	W	Zeichenzeilen
00111	R7	VSYNC Position	W	Zeichenzeilen
01000	R8	Interlace (Zeilensprung)	W	–
01001	R9	Anzahl der Zeilen pro Zeichen	W	Bildzeilen
01010	R10	Cursor start	W	Bildzeile
01011	R11	Cursor end	W	Bildzeile
01100	R12	Speicheradr. Start H	W	–
01101	R13	Speicheradr. Start L	W	–
01110	R14	Cursor H	RW	–
01111	R15	Cursor L	RW	–
10000	R16	Lichtgriffel Adr. H	R	–
10001	R17	Lichtgriffel Adr. L	R	–

Es ist natürlich voll TTL kompatibel und kann, obwohl für das Motorola System 6800 ausgelegt, für die meisten Prozessoren direkt verwendet werden.

Abb. 1.3.1.1-3 zeigt die PIN-Belegung des CRT-Controlers 6845. Der Baustein belegt nur zwei IO-Adressen, zur Auswahl dient der Eingang RS (register select).

Intern besitzt der 6845 17 adressierbare Register. Die Auswahl eines dieser Register erfolgt dadurch, daß die Registeradresse auf die Portadresse 0 (RS = 0) geschrieben wird und dann können auf der Portadresse 1 (RS = 1) Daten in dieses Register geschrieben werden, oder daraus gelesen werden.

Tabelle 1.3.1.1-3 gibt einen Überblick von den einzelnen Funktionen der Register und *Tabelle 1.3.1.1-4* zeigt die genaue Belegung der einzelnen Bits.

Neben dem linearen Adreßgenerator bietet das IC auch noch die Möglichkeit, verschiedene Displayvarianten zu programmieren, sowie unterschiedliche Cursorarten zu wählen.

Abb. 1.3.1.1-4 zeigt die verschiedenen Möglichkeiten zur Einstellung des Displayverfahrens.

Dabei kann zwischen dem Zeilensprungverfahren und dem 50-Hz-Bildwechsel gewählt werden.

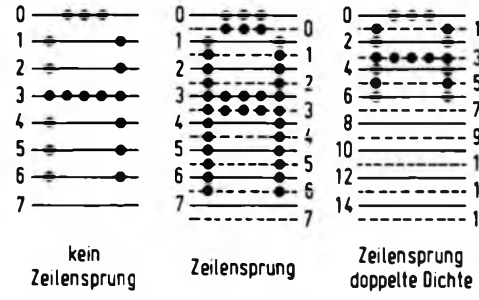
Beim Zeilensprungverfahren wird zwischen zwei Darstellungsarten unterschieden. Einmal können die Zeichen in der gleichen Größe belassen werden, wie beim 50-Hz-Bildwechsel ohne Zeilensprung, oder es kann die doppelte Menge an Zeichen in vertikaler Richtung dargestellt werden.

Tabelle 1.3.1.1-4 a

Register	Bezeichnung	7 6 5 4 3 2 1 0
0	Hier wird mit 8 Bits die gesamte Zahl der Zeichen angegeben, wobei auch die nicht dargestellten gezählt werden. Der Wert ergibt sich aus der Zahl – 1. Mit dieser Zahl wird die Horizontale Wiederholfrequenz festgelegt.	
1	Hier werden mit 8 Bits die Zahl der dargestellten Zeichen angegeben, bezogen auf eine Zeile.	7 0
2	Mit 8 Bits dieses Registers kann die Position des HSYNC Signals bestimmt werden.	7 0
3	Mit 4 Bits wird die Breite des HSYNC Signals eingestellt.	3 0
4	Mit R4 kann die Bildwiederholrate (50 Hz oder 60 Hz) bestimmt werden. Dazu wird mit 7 Bits die Anzahl der Zeichenzeilen gesamt angegeben.	6 0
5	Mit 5 Bits dieses Registers kann der Feinabgleich in Bildzeilen durchgeführt werden, um auf die richtige Bildwiederholrate zu kommen.	
6	Mit 7 Bits kann die Anzahl der dargestellten Zeichenzeilen bestimmt werden.	6 0
7	Mit 7 Bits dieses Registers wird die VSYNC Position eingestellt. Dies geschieht in Zeichenzeilen.	6 0
8	Mit diesem Register kann in 2 Bits der Display Mode eingestellt werden, der bestimmt, ob im Zeilensprungverfahren gearbeitet wird.	1 0
9	Mit 5 Bits dieses Registers kann die Anzahl der Bildzeilen für ein Zeichen eingestellt werden (RA0 bis RA4).	4 0
10	Bit 0 bis 4 bestimmen die Zeilenstartadresse des Cursors. Bit 5 und 6 bestimmen den Cursor Modus (nicht blinkend, nicht dargestellt, blinkend mit 1/16 der vertikalen Feldrate, oder mit 1/32 der vert. Feldrate.	6 5 4 0
11	Bit 0 bis 4 bestimmen die Zeilenendadresse des Cursors.	
12	Hier wird der höherwertige Teil der Startadresse des Speichers angegeben, bei dem die Darstellung erfolgen soll. Dazu dienen 6 Bits.	5 0
13	Low Adresse für die Speicherstartadresse. 8 Bits.	7 0
14	High Adresse, des Starts für die Cursorposition, die direkt einer Speicheradresse zugeordnet ist. 6 Bits.	5 0
15	Low Adresse mit 8 Bits für die Cursorposition.	7 0
16	Es wird die High Adresse (6 Bits) festgehalten, wenn der LPSTB Eingang einen Puls erhält, die mit der Speicheradresse des gerade dargestellten Zeichens übereinstimmt.	5 0
17	8 Bits für die Low Adresse des LPSTB Registers.	7 0

Abb. 1.3.1.1-4 Verschiedene Darstellungsarten von Zeichen

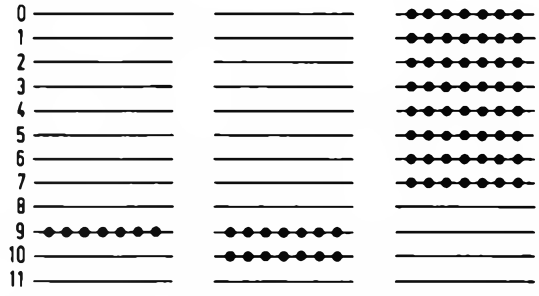
Bit 1	Bit 0	Modus
0 1	0 0	kein Zeilensprung
0	1	Zeilensprung
1	1	Zeilensprung / doppelte Dichte



Cursor Anzeige Modus

Bit 6	Bit 5	Cursor Modus
0	0	nicht blinkend
0	1	nicht dargestellt
1	0	1/16 Feldrate blinkend
1	1	1/32 Feldrate blinkend

Cursor Format



Cursor Start	9	9	0
Cursor End	9	10	7

Abb. 1.3.1.1-5 Einstellmöglichkeiten bei der Cursorwahl

Der Nachteil des Zeilensprungverfahrens ist, daß eine effektive Bildwechselfrequenz von 25 Hz zustande kommt und bei gewöhnlichen TV-Geräten oder Monitorgeräten ein Flimmereffekt wirksam wird. Abhilfe kann da nur ein Monitor mit nachleuchtender Bildröhre schaffen.

Abb. 1.3.1.1-5 zeigt die Einstellmöglichkeiten bei der Cursorwahl.

Der Cursor kann blinkend oder nichtblinkend dargestellt werden. Ferner kann die Höhe und Lage des Cursors eingestellt werden, damit ist es auch möglich, den Cursor unterhalb der Zeichengrenze zu legen, um ihn besser unterscheiden zu können.

Abb. 1.3.1.1-6 zeigt den Anschluß an die Z80-CPU. Der Datenbus ist wie üblich über den Bustreiber B1 gepuffert und die Adreßselektion wird mit der Vergleicherlogik V1 und V2 vorgenommen. Die Adresse läßt sich mit J1 bis J7 einstellen.

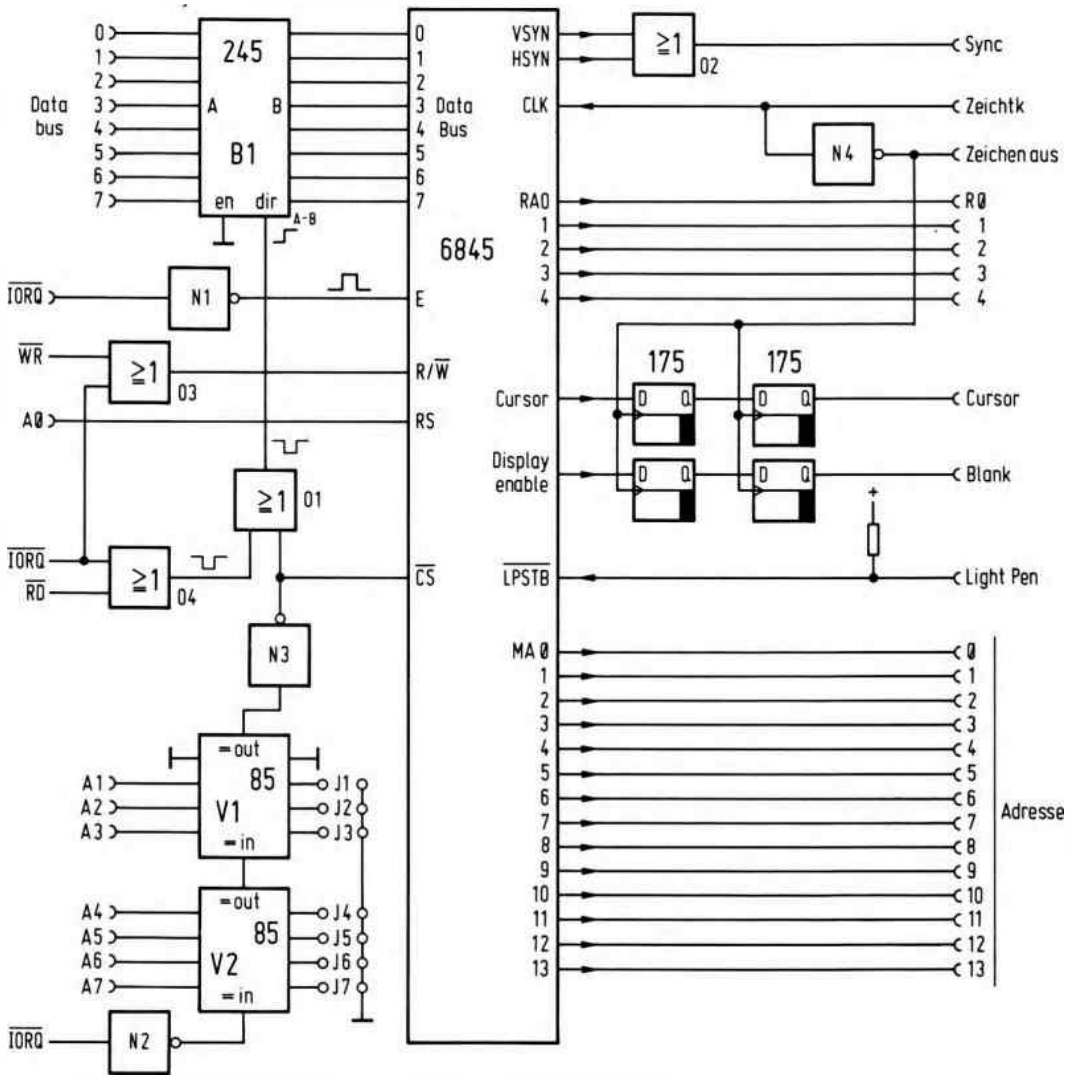


Abb. 1.3.1.1-6 Anschluß des 6845 an den Z80

```
                        .PABS
E000                    .LOC 0E000H
                        .PHEX
                        ;
                        ;
                        ;************************
                        ;* KLEINES CRT DISPLAY *
                        ;* PROGRAMM 791030     *
                        ;* RDK      1.0        *
                        ;************************

E000    C3 E00C         JMP INIT
E003    C3 E00C         JMP INIT
E006    C3 E00C         JMP INIT
E009    C3 E036         JMP CO  ;09 AL=U
E00C                    INIT:
E00C    21 E01D         LXI H,TAB
E00F                    ..LP:
E00F    7E              MOV A,M

E010    FEFF            CPI 0FFH        ;ENDE DER TABELLE
E012    CA E030         JZ CONT
E015    4F              MOV C,A
E016    23              INX H
E017    7E              MOV A,M
E018    ED79            OUTP A
E01A    23              INX H
E01B    18F2            JMPR ..LP
E01D                    TAB:
E01D    0051            .BYTE 0,51H     ;HORIZ. LINES
E01F    011D            .BYTE 1,1DH     ;SYNC ,NON INTERLANCE
E021    0238            .BYTE 2,38H     ;64 ZEICHEN,16 SCANS
E023    03DF            .BYTE 3,0DFH    ;SKEW=3,32 ZEICHEN ZEILEN
E025    0410            .BYTE 4,10H     ;SCANS/FRAME
E027    0518            .BYTE 5,18H     ;VERTIKAL DATA START
E029    061F            .BYTE 6,1FH     ;STOPP BEI 32 ZEICHEN
E02B    0C06            .BYTE 0CH,6     ;CURSOR
E02D    0D00            .BYTE 0DH,0     ;CURSOR
E02F    FF              .BYTE 0FFH      ;ENDE DER TABELLE
                        ;
E030                    CONT:
E030    CD E08C         CALL CLEAR
E033    CD F01E         CALL 0F01EH     ;MONITOR
                        ;
                        ;
E036                    CO:
E036    79              MOV A,C         ;EINGABEZEICHEN
E037    FE0D            CPI 0DH         ;CR
E039    CA E067         JZ CR
E03C    FE0A            CPI 0AH
E03E    CA E0A7         JZ LF
E041    FE0C            CPI 0CH
E043    CA E08C         JZ CLEAR
E046    E5              PUSH H
E047    2A E0F2         LHLD POSITION
E04A    71              MOV M,C         ;ABSPEICHERN ZEICHEN
E04B    23              INX H
E04C    22 E0F2         SHLD POSITION
E04F    7C              MOV A,H
E050    FEDF            CPI 0DFH        ;ENDE BEREICH
E052    C2 E061         JNZ ENDE
```

Abb. 1.3.1.1-7 Programm zur Steuerung des CRT-Displays

```
E055    7D          MOV A,L
E056    FEFF        CPI 0FFH
E058    C2 E061     JNZ ENDE
E05B    21 D800     LXI H,0D800H
E05E    22 E0E2     SHLD POSITION
E061                ENDE:
E061    E1          POP H
E062    79          MOV A,C
E063    CD E0C5     CALL CURSOR
E066    C9          RET
                    ;
                    ;
E067                CR:
E067    E5          PUSH H                          zu Abb. 1.3.1.1-7
E068    2A E0E2     LHLD POSITION
E06B    7D          MOV A,L
E06C    FE00        CPI 0
E06E    CA E083     JZ CONI2
E071                ..LP1:
E071    7D          MOV A,L
E072    E63F        ANI 03FH
E074    2805        JRZ CONI
E076    3620        MVI M,20H       ;LOESCHE RESTZEILE
E078    23          INX H
E079    18F6        JMPR ..LP1
E07B                CONI:
E07B    D5          PUSH D
E07C    11 0040     LXI D,40H
E07F    AF          XRA A
E080    ED52        DSBC D
E082    D1          POP D
E083                CONI2:
E083    22 E0E2     SHLD POSITION
E086    E1          POP H
E087    79          MOV A,C
E088    CD E0C5     CALL CURSOR
E08B    C9          RET
                    ;
                    ;
E08C                CLEAR:
E08C    E5          PUSH H
E08D    D5          PUSH D
E08E    C5          PUSH B
E08F    21 D800     LXI H,0D800H
E092    22 E0E2     SHLD POSITION
E095    3620        MVI M,20H
E097    11 D801     LXI D,0D800H+1
E09A    01 07FF     LXI B,07FFH
E09D    EDB0        LDIR
E09F    C1          POP B
E0A0    D1          POP D
E0A1    E1          POP H
E0A2    79          MOV A,C
E0A3    CD E0C5     CALL CURSOR
E0A6    C9          RET
                    ;
E0A7                LF:
E0A7    E5          PUSH H
E0A8    2A E0E2     LHLD POSITION
E0AB    D5          PUSH D
E0AC    11 0040     LXI D,40H
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 3
.MAIN. - CRT PROGRAMM


   E0AF    19            DAD D
   E0B0    7C            MOV A,H
   E0B1    FEE0          CPI 0E0H
   E0B3    C2 E0BB       JNZ CONI1
   E0B6    CD E0BC       CALL CLEAR
   E0B9    1803          JMPR FERTIG
   E0BB                  CONI1:
   E0BB    22 E0E2       SHLD POSITION
   E0BE                  FERTIG:
   E0BE    D1            POP D
   E0BF    E1            POP H                    zu Abb. 1.3.1.1-7
   E0C0    79            MOV A,C
   E0C1    CD E0C5       CALL CURSOR
   E0C4    C9            RET
                         ;
                         ;
   E0C5                  CURSOR:
   E0C5    F5            PUSH PSW
   E0C6    E5            PUSH H
   E0C7    2A E0E2       LHLD POSITION
   E0CA    7D            MOV A,L
   E0CB    E63F          ANI 3FH
   E0CD    D30C          OUT 0CH          ;CURSOR HORIZ.
   E0CF    7D            MOV A,L
   E0D0    E6C0          ANI 0C0H
   E0D2    07            RLC
   E0D3    07            RLC
   E0D4    C5            PUSH B
   E0D5    47            MOV B,A
   E0D6    7C            MOV A,H
   E0D7    E607          ANI 7
   E0D9    07            RLC
   E0DA    07            RLC              ;000XXX00
   E0DB    B0            ORA B            ;000XXXYY
   E0DC    D30D          OUT 0DH          ;VERTIKAL CURSOR POS.
   E0DE    C1            POP B
   E0DF    E1            POP H
   E0E0    F1            POP PSW
   E0E1    C9            RET
                         ;
   E0E2    0000          POSITION: .WORD 0
                         .END
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. - CRT PROGRAMM
+++++ SYMBOL TABLE +++++

CLEAR  E08C          CO     E036          CONI   E07B          CONI1  E0BB
CONI2  E083          CONT   E038          CR     E067          CURSOR E0C5
ENDE   E061          FERTIG E0BE          INIT   E08C          LF     E0A7
POSITI E0E2          TAB    E01D          .BLNK. 0000:03 X     .DATA. 0000*   X
.PROG. 0000'   X
```

Die Richtungssteuerung erfolgt über das Gatter 01, dessen Ausgang immer dann auf Low geht, wenn die Adresse stimmt und ein Lesezugriff vorliegt. Das Signal R/W wird immer dann erzeugt, wenn ein Schreibzugriff vorliegt. Der Eingang $\overline{CS}$ erhält einen Low-Pegel, wenn eine gültige Adresse anliegt und mit dem Eingang E wird bei einem IORQ die Gültigkeit angezeigt. Der Eingang RS (register select) ist direkt mit der Adreßleitung A0 verbunden, so daß auf der niederwertigen Adresse das Adreßregister angesprochen wird und auf der höherwertigen, das jeweils adressierte Register. Der CRT-Controler erzeugt eine Ausleseadresse, die an den Ausgängen MA0 bis MA13 anliegt. Damit können bis zu 16K Bildwiederholspeicher adressiert werden.

Das SYNC-Signal wird mit einem ODER-Gatter aus den Signalen VSYN und HSYN gewonnen. Der Eingang CLK wird mit dem Zeichentakt verbunden und bestimmt den Ablauf der Displaysteuerung. Das Signal CLK wurde mit dem Gatter N4 invertiert und führt an einen Ausgang mit dem Namen „Zeichen aus", um mit der Schaltung des CRT-Controlers 5027 kompatibel zu bleiben. Die Ausgänge R0 bis R4 dienen der Steuerung eines Zeichengenerators und bestimmen die Zeile innerhalb eines Zeichens. Die Ausgänge CURSOR und BLANK müssen mit je zwei Flipflops, die als Schieberegister geschaltet sind, um zwei Zeichentakte verschoben werden, um mit der nachfolgenden Speicherschaltung kompatibel zu sein. Diese Verschiebung war beim 5027 nicht notwendig, da sie dort per Software programmiert werden konnte.

Der Eingang $\overline{LPSTB}$ dient dem Anschluß eines Lichtgriffels, der einen Impuls liefern muß, sobald der Schreibstrahl des TV-Gerätes an der Stelle steht, an der sich auch der Lichtgriffel befindet. Dann wird in einem internen Register die gerade anstehende Adresse des Bildwiederholspeichers abgelegt und kann später von der CPU ausgelesen werden.

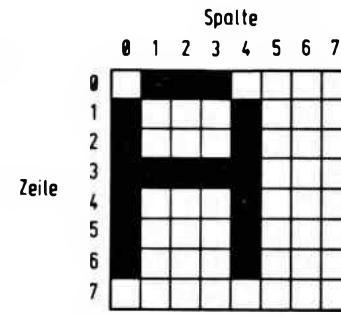
1.3.1.2 Anschluß von Zeichengeneratoren und Graphikschaltungen an den CRT-Controler

Im vorherigen Kapitel wurden die CRT-Controler 6845 und 5027 vorgestellt. Hier sollen nun verschiedene Schaltungen besprochen werden, die an die CRT-Schaltung angeschlossen werden und die eigentliche Zeichendarstellung bestimmen. Prinzipiell muß bei der Auswahl der hier vorgestellten Schaltungen entschieden werden, ob bei dem Datensichtgerät nur alphanumerische Zeichen dargestellt werden sollen, oder ob es möglich sein soll, auch Graphik-Darstellungen durchzuführen.

Bei der ausschließlichen Verwendung von alphanumerischen Zeichen ist die Schaltung relativ einfach. *Abb. 1.3.1.2-1* zeigt den Aufbau eines Zeichens, wie es von einem Zeichengenerator vorgenommen wird (z.B. mit 2708). Beim 2708 steht ein Raster von 8 x 8 Bildpunkten zur Verfügung, das auch zur Darstellung von Kleinbuchstaben mit Unterlängen geeignet ist. Auch ist eine beschränkte Graphik-Möglichkeit vorhanden, wenn vorgefertigte Graphikelemente ähnlich wie bei PET programmiert werden. *Abb. 1.3.1.2-2* zeigt die mögliche Realisierung eines Zeichensatzes, wie er in der SIEMENS HOBBY GRUPPE MHG (nur für Siemens Angehörige) verwendet wird. Dieses Muster kann nach einer Binärumschlüsselung direkt in den 2708 übernommen werden.

Abb. 1.3.1.2-3 zeigt die Ansteuerschaltung mit dem Zeichengenerator. Die Schaltung wird mit einem Takt von 10 MHz betrieben, um damit eine Darstellung 64 Zeichen/

Abb. 1.3.1.2-1 Aufbau eines Zeichens



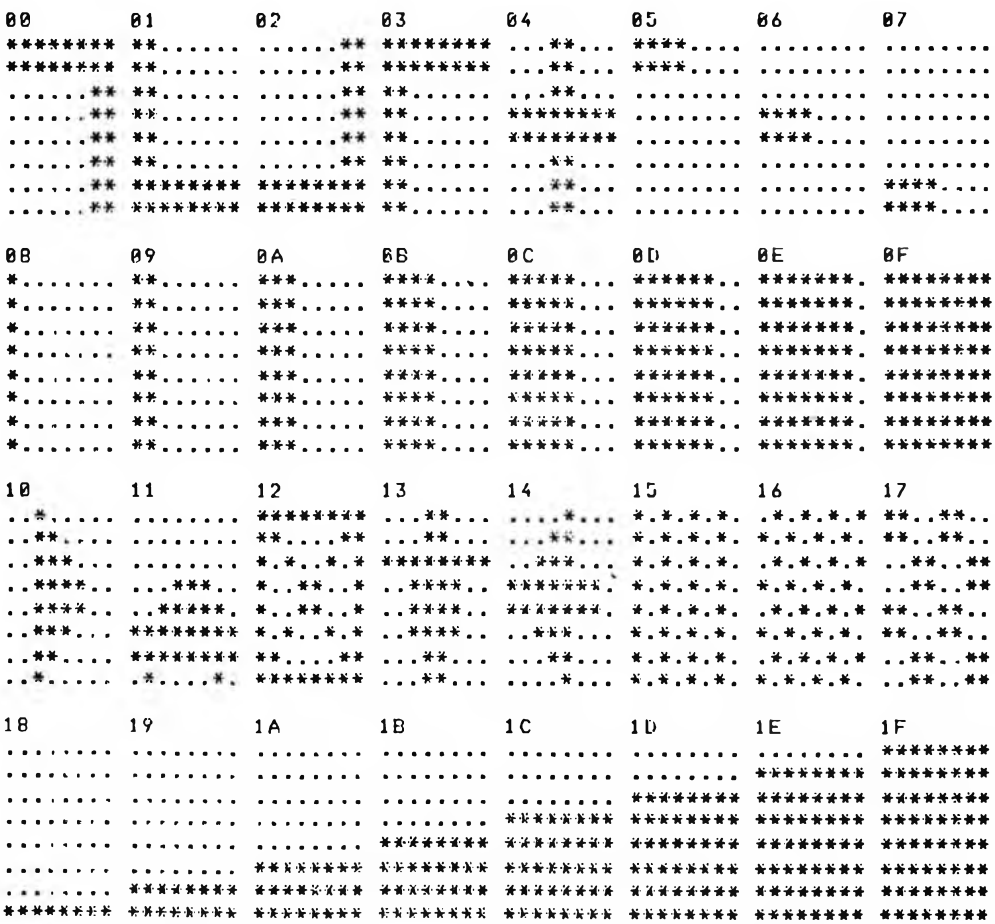
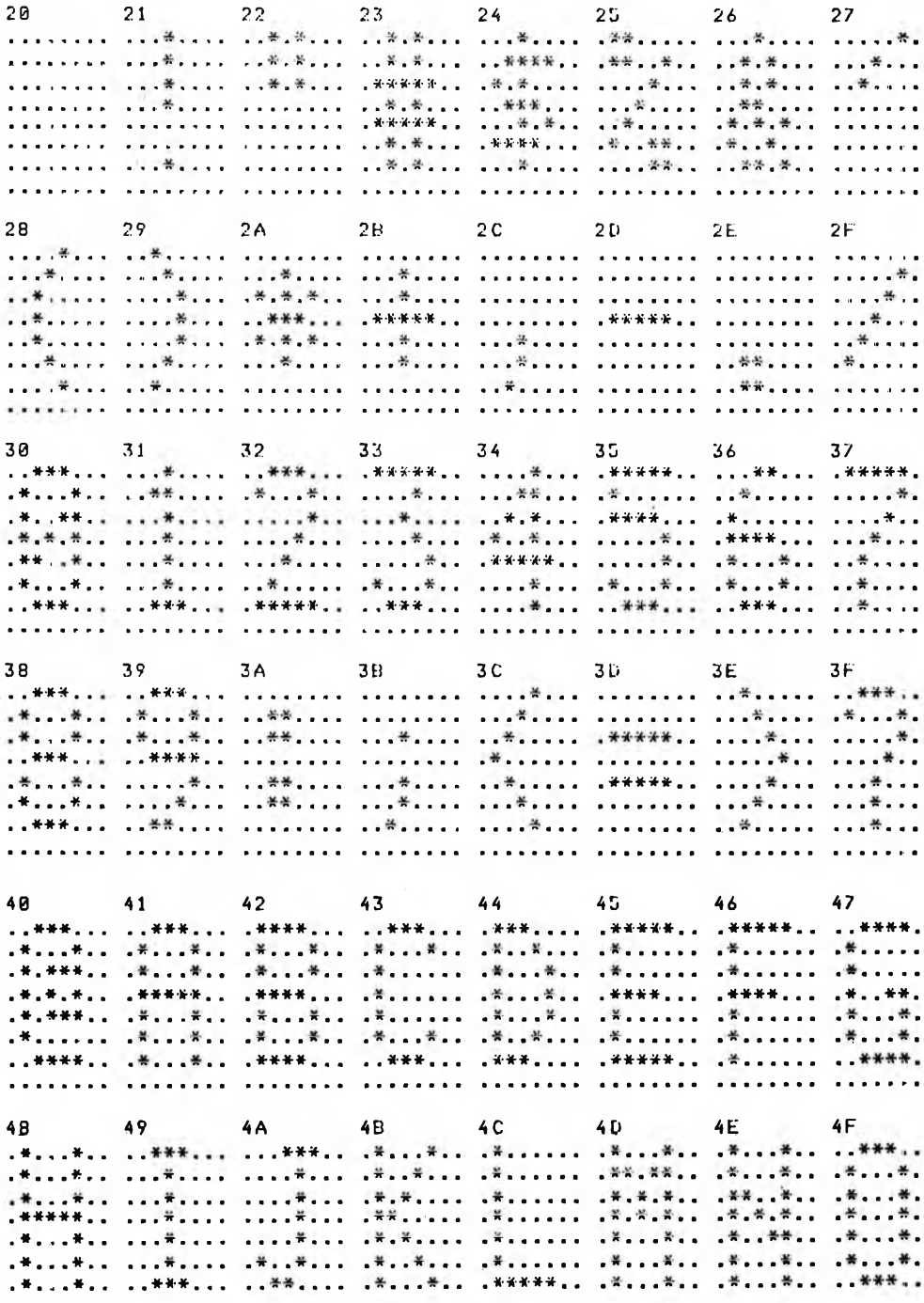
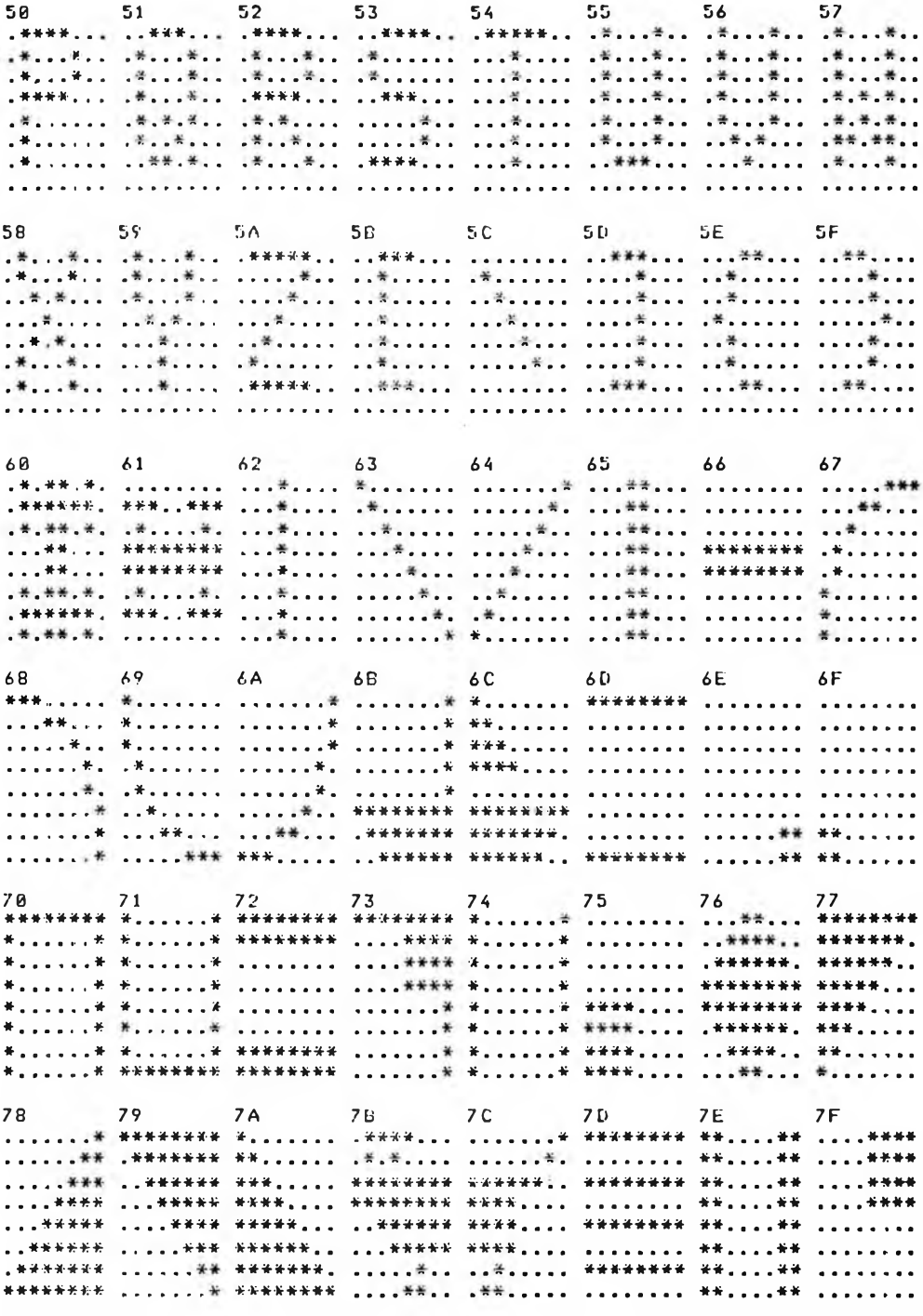


Abb. 1.3.1.2-2 Beispiel eines Zeichensatzes (SIEMENS HOBBY GRUPPE)


20
21
22
23
24
25
26
27
28
29
2A
2B
2C
2D
2E
2F
30
31
32
33
34
35
36
37
38
39
3A
3B
3C
3D
3E
3F
40
41
42
43
44
45
46
47
48
49
4A
4B
4C
4D
4E
4F

zu Abb. 1.3.1.2-2


50 51 52 53 54 55 56 57
58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67
68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77
78 79 7A 7B 7C 7D 7E 7F

zu Abb. 1.3.1.2-2

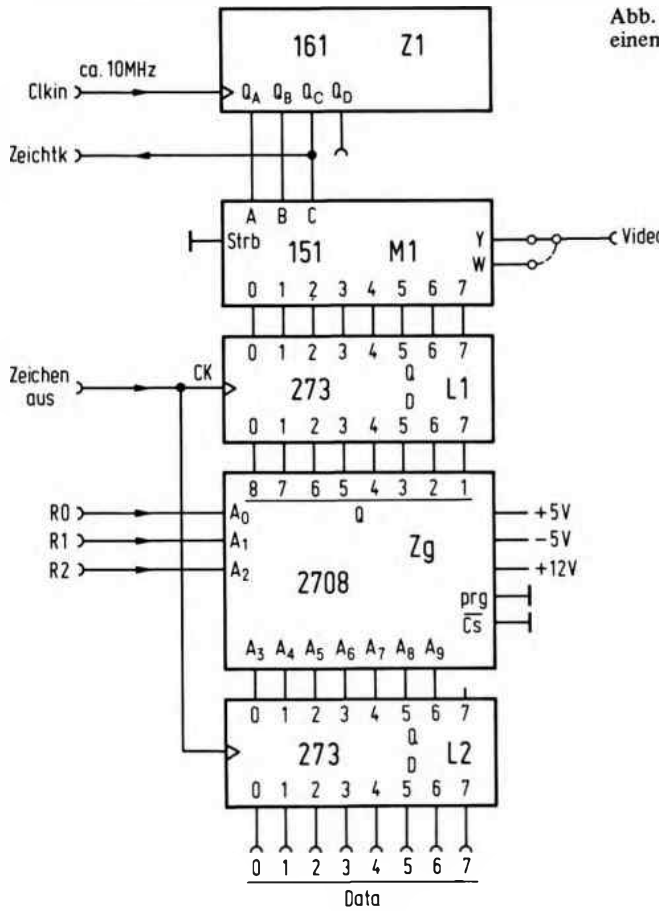


Abb. 1.3.1.2-3 Ansteuerschaltung mit einem Zeichengenerator

Zeile zu ermöglichen. Der 10 MHz-Takt gelangt an den Eingang des Zählers Z1, der die Punktposition innerhalb eines Zeichens in horizontaler Richtung bestimmt. Dabei wird ein Zeichen mit acht Punkten dargestellt. Der Zwischenraum zwischen zwei Zeichen ist durch die Programmierung des Zeichengenerators bestimmt. Der Ausgang „Zeichentk" wird an den CRT-Controler geführt. Der Eingang „Zeichen aus" führt ebenfalls den Zeichentakt, er wird von der CRT-Schaltung gewonnen und ist gegenüber „Zeichentk" je nach Controler ggf. invertiert. Die Ausgänge QA bis QC sind mit einem Multiplexer verbunden. Der Multiplexer hat die Aufgabe, einen Punkt auszuwählen und an den BAS-Mischer weiterzuleiten.

Die acht Eingänge des Multiplexers sind mit dem Zwischenspeicher L1 verbunden. Dieser Zwischenspeicher arbeitet wie ein Schieberegister und muß die ankommende Information um eine Zeichenbreite verzögern. Dies ist nötig, da die Zugriffszeit des EPROMs mit 450 ns in gleicher Größenordnung wie die Zeichenbreite liegt, und daher ein Zeichen nicht rechtzeitig zur Verfügung stehen könnte. Mit diesem Pipelineverfahren kann das Problem umgangen werden. Der Zeichengenerator erhält die Adresse einmal

direkt vom CRT-Controler (R0 bis R2) und dann noch vom Ausgang des Zwischenspeichers L2. Auch hier muß der Zeitkonflikt, der sich aus der großen Zugriffszeit des Bildwiederholspeichers ergibt, beseitigt werden.

Wird eine Zeilenzahl pro Zeichen gewählt, die größer als 8 ist, so arbeitet die Schaltung nicht korrekt, da R3 und R4 vom CRT-Controler durch diese Schaltung nicht ausgewertet wird. Wird also zum Beispiel Zeile 9 adressiert, so gibt der Zeichengenerator das Muster für Zeile 1 aus. Es ist daher nötig, die Signale R3 und R4 mit dem Videosignal zu verknüpfen. Die Verbindung mit $\overline{CS}$ funktioniert nicht, da dann am Ausgang des EPROMs die Signale 1 (high impedance bei TTL) erscheinen und damit das Zeichen hell erscheint. Eine Möglichkeit besteht, den EPROM z.B. invers zu programmieren. Eine vom Zeichengenerator unabhängige Methode ist die Verknüpfung hinter dem Multiplexer, z.B. mit dem Zeichensatz, der in Abb. 1.3.1.2-2 gezeigt wurde, ist eine beschränkte Möglichkeit zur Graphik-Darstellung gegeben. Soll eine höhere Auflösung möglich sein, so kann eine der nachfolgenden Schaltungen mit der alphanumerischen kombiniert oder getrennt verwendet werden. Eine Kombinierung ist zum Beispiel durch ein zusätzliches Bit im Bildwiederholspeicher möglich, das die Auswahl des Videosignals steuert.

Abb. 1.3.1.2-4 zeigt die Aufteilung eines Bytes im Bildspeicher in ein Graphikfeld. Die Auflösung ist relativ gering, weshalb diese Darstellungsart im amerikanischen Jargon auch als „sparse graphic" bezeichnet wird. Ein Graphikfeld besteht aus 8 x 8 Bildpunkten, wobei ein adressierbares Feld aus 4 x 2 Punkten besteht. *Abb. 1.3.1.2-5* zeigt

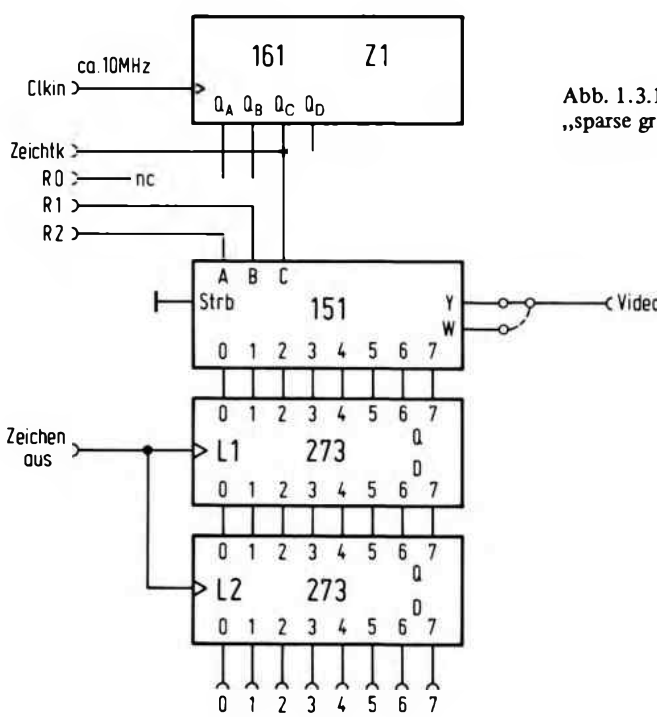


Abb. 1.3.1.2-5 Schaltung zur Erzeugung von „sparse graphic"

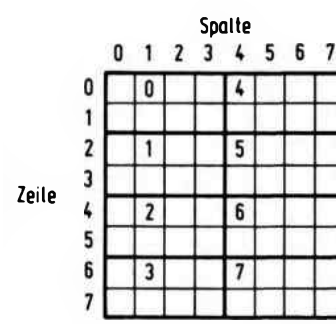


Abb. 1.3.1.2-4 „Sparse graphic"

die dazugehörige Schaltung. Sie ist ähnlich zu der Schaltung für alphanumerische Darstellung, nur daß der Zeichengenerator entfällt und die Zuordnung der Auswahlleitungen des Multiplexers anders ist. Die Eingänge A und B des Multiplexers führen direkt an R1 und R2 der CRT-Schaltung. Damit wird die Bildfläche in vier Zeilen unterteilt, wenn der CRT-Controler für acht Zeilen programmiert wurde.

Der Eingang C des Multiplexers führt an den Ausgang QC des Zählers Z1. Dadurch wird ein Zeichen in horizontaler Richtung in zwei Hälften zerteilt, mit gesamt acht Bildpunkten. Die doppelte Verzögerung mit L1 und L2 wäre hier eigentlich nicht notwendig, da die Zugriffszeit des Zeichengenerators entfällt, sie wurde aber dennoch vorgesehen, um diese Schaltung mit der vorhergehenden kombinieren zu können.

Diese Graphik-Schaltung kommt mit einem relativ kleinen Bildwiederholspeicher aus. Bei einem Feld von 64 x 32 Zeichen, also einer Auflösung von 128 x 128 adressierbaren Punkten, wird nur ein Speicher von 2K Bytes benötigt. Eine etwas höhere Auflösung wird mit der sogenannten „dense graphic" erreicht. *Abb. 1.3.1.2-6* zeigt die Aufteilung

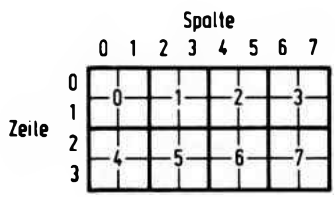


Abb. 1.3.1.2-6 „dense graphic"

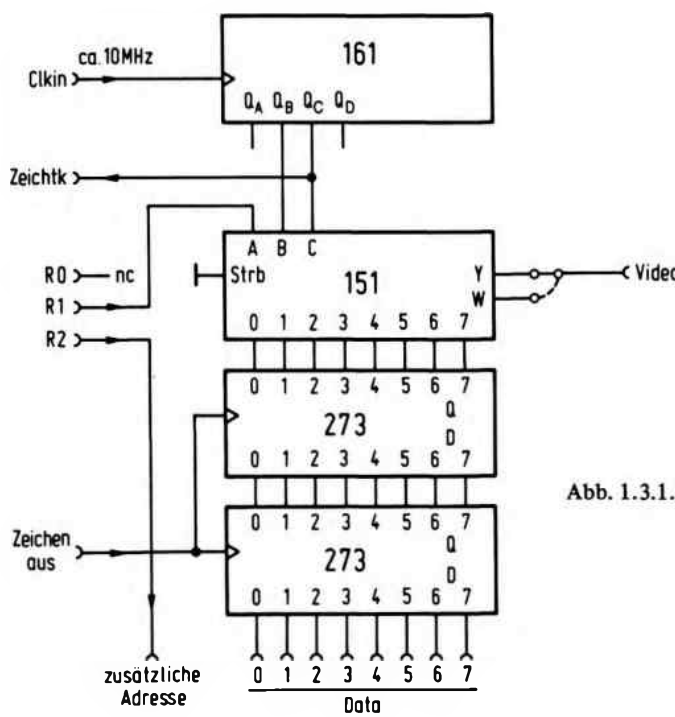


Abb. 1.3.1.2-7 Schaltung für „dense graphic"

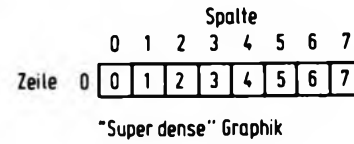


Abb. 1.3.1.2-8 „super dense graphic"

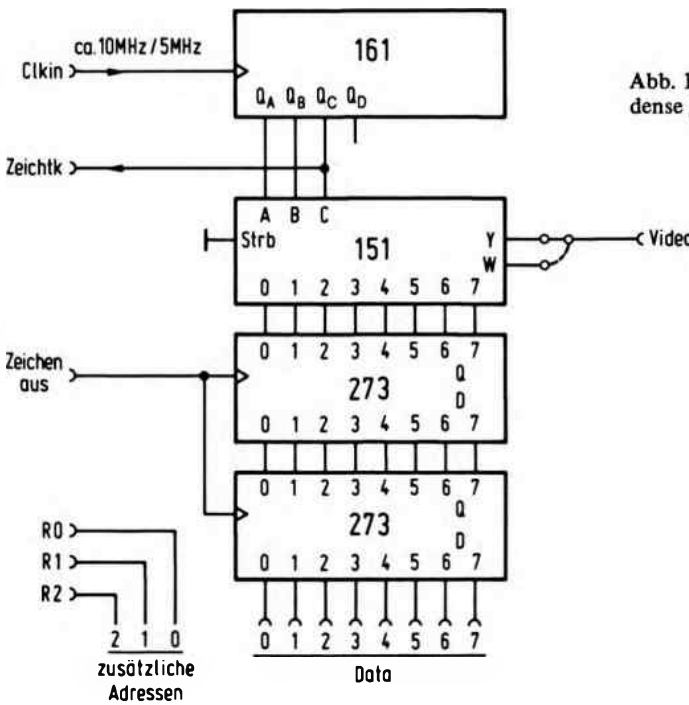


Abb. 1.3.1.2-9 Schaltung zur „super dense graphic"

des Bildfeldes für diese Darstellungsart. Pro Feld sind 8 x 4 Bildpunkte verwendet, wobei 4 x 2 Bildpunkte adressierbar sind, also jeweils vier Bildpunkte zu einem adressierbaren Bildpunkt zusammengefaßt werden. *Abb. 1.3.1.2-7* zeigt die Schaltung. Sie ist fast identisch mit der Schaltung für den „sparse dense"-Mode, nur ist die Aufteilung der Multiplexersteuerung anders. Für den Bildwiederholspeicher muß nun eine zusätzliche Adresse zur Verfügung gestellt werden, da für die höhere Auflösung auch ein größerer Bildwiederholspeicher nötig ist.

Die Adresse wird aus dem Signal R2 des CRT-Controlers gebildet. Es ist aber auch anders möglich, diese zusätzliche Adresse zu gewinnen. Wird beim 6845 z.B. eine Zeichenhöhe von vier Bildzeilen gewählt und die Anzahl der Zeichen in vertikaler Richtung verdoppelt, so entsteht die zusätzliche Adresse automatisch an den Leitungen MA0 bis MA13. Die Schaltung würde sich sonst nicht weiter verändern.

Abb. 1.3.1.2-8 zeigt den Aufbau eines Feldelementes für die „super dense graphic". Jetzt ist jeder Bildpunkt adressierbar. *Abb. 1.3.1.2-9* zeigt die entsprechende Schaltung.

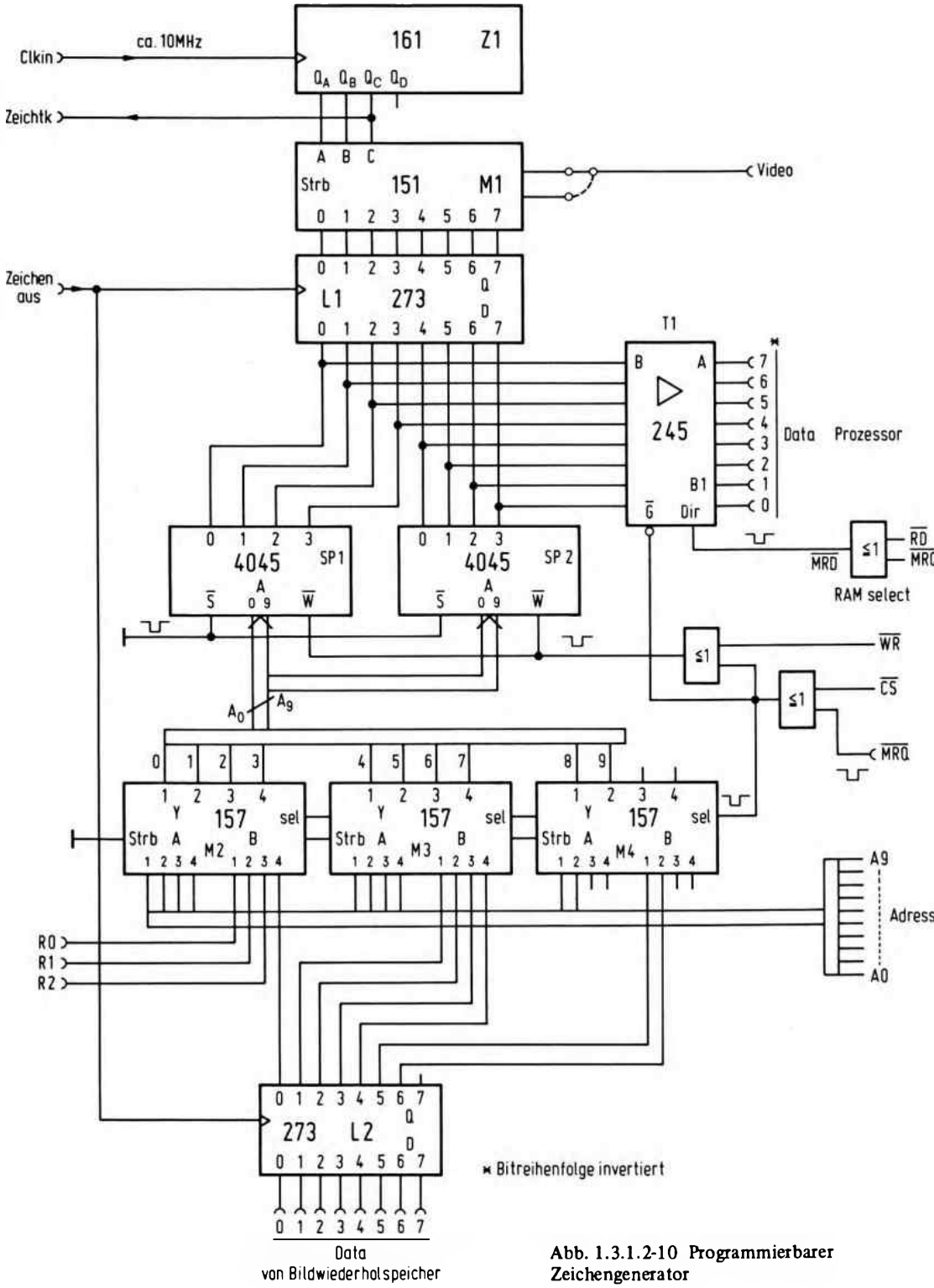


Abb. 1.3.1.2-10 Programmierbarer Zeichengenerator

Der Bildwiederholspeicher benötigt nun eine Kapazität von 8K Bytes, und es müssen gegenüber der Schaltung für die Darstellung von alphanumerischen Zeichen drei Adressen für den Bildwiederholspeicher hinzugefügt werden. Eine Reduzierung in horizontaler Richtung auf 32 Zeichen ist dabei vorausgesetzt, sonst wären 16K Bytes nötig. Es wird also eine Auflösung von 32 x 8 mal 32 x 8, also 256 x 256 Bildpunkten erreicht, die getrennt adressierbar sind. Bei dieser Auflösung ist es möglich, alphanumerische Zeichen direkt durch Einschreiben des Punktmusters darzustellen, wobei diese auch noch beliebig verschoben werden können. Der Zeichengenerator ist dann in einem Programm als Tabelle ausgeführt. Eine weitere Möglichkeit, graphische Darstellung zu realisieren, ist die Verwendung eines programmierbaren Zeichengenerators. Dabei wird anstelle des EPROMs, das den Zeichensatz enthält, ein RAM verwendet. Der Prozessor kann nun den Zeichensatz während des Programmverlaufs ändern und so für eine bestimmte Aufgabe vorbereiten, für die nur bestimmte Graphik-Grundelemente verwendet werden. Damit ist es möglich, den hohen Speicherbedarf für eine „super dense"-Graphik einzusparen und trotzdem die gleiche Auflösung zu erhalten, ohne an einen festen Graphiksatz gebunden zu sein. Diese Methode ist natürlich nur ein begrenzter Ersatz für eine hochauflösende Graphik, doch kann in den meisten Fällen, wie z.B. Schachfiguren, Digitalschaltungen, Layouts, Prozeßablaufdiagramme, APL und andere Spezialschriftsätze mit einem programmierbaren Zeichengenerator eine ausreichende Lösung gefunden werden. *Abb. 1.3.1.2-10* zeigt die dazugehörige Schaltung. Sie ist praktisch genauso aufgebaut, wie die Schaltung zur alphanumerischen Darstellung in Abb. 1.3.1.2-3, nur daß anstelle des EPROMs eine Schaltung mit RAMs getreten ist. Es werden zwei RAM-Bausteine mit einer Organisation von 1024 x 4 verwendet. Die Adreßeingänge der RAMs sind mit den Ausgängen von Multiplexern verbunden, die im Normalfall die Adressen von R0 bis R3 und dem Ausgang des Latches L2 (als Schieberegister) durchschalten. Soll ein Zeichensatz programmiert werden, so werden die Adressen vom CPU-Adreßbus an den Speicher geleitet.

Die Datenein-/ -ausgänge des Speichers gelangen an den Eingang von L2 und an die B-Seite des bidirektionalen Bustreibers B1. Die Multiplexer und der Bustreiber werden mit den Signalen $\overline{CS}$ und $\overline{MRQ}$ gesteuert. Nur wenn beide Signale auf Low sind, wird der Bustreiber freigegeben und die Multiplexer werden auf die A-Seite umgeschaltet. Das Signal $\overline{CS}$ kann zum Beispiel mit der üblichen Vergleicherschaltung gewonnen werden. Liegt ein Lesezugriff vor, so wird der Bustreiber in der Richtung B nach A geschaltet.

Das Schreibsignal wird an den Speicher gelegt, sobald $\overline{CS}$, $\overline{MRQ}$ und $\overline{WR}$ auf Low liegen, also aktiv sind. Von der CPU aus gesehen, erscheint die Schaltung wie ein gewöhnlicher Speicher.

1.3.1.3 Restliche Baugruppen für ein CRT-Gerät

Abb. 1.3.1.3-1 zeigt das Blockschaltbild eines gesamten Datensichtgerätes. Es besteht aus fünf verschiedenen Modulen, von denen die CRT-Steuerung und der Zeichengenerator schon beschrieben wurden. Die restlichen Baugruppen, RAM, Mischer und HF-Stufe stehen noch aus. Die RAM-Baugruppe ist der Bildwiederholspeicher und da sie von zwei verschiedenen Systemen (CRT und CPU) aus adressierbar sein muß, ergibt dies einen besonderen Aufbau.

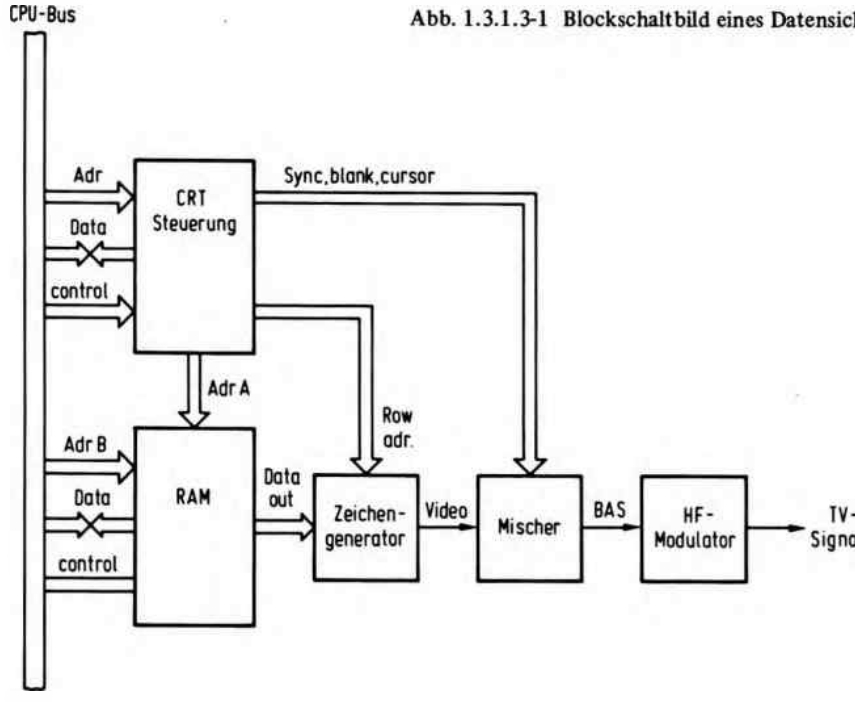


Abb. 1.3.1.3-1 Blockschaltbild eines Datensichtgerätes

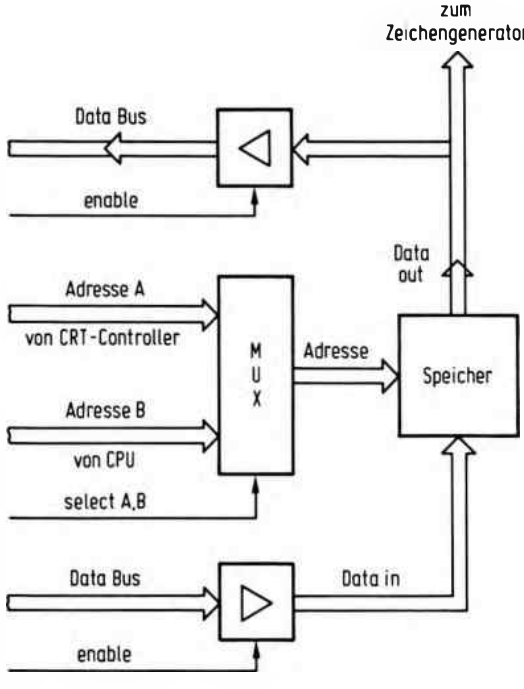


Abb. 1.3.1.3-2 Bildwiederholspeicher

1. Bildwiederholspeicher

Abb. 1.3.1.3-2 zeigt das Blockschaltbild eines Bildwiederholspeichers. Er besteht aus dem eigentlichen Speicher, einem Multiplexer und Treibern. Der Multiplexer hat ähnlich wie beim programmierbaren Zeichengenerator die Aufgabe, im Normalfall die Adressen der CRT-Steuerung an das RAM weiterzuleiten, und nur, falls die CPU Daten in den Bildwiederholspeicher einschreiben will, die Adressen der CPU durchzuschalten. Es ergeben sich hier aber zusätzliche Probleme. Wenn der Prozessor auf den Bildwiederholspeicher zugreift, so kann für die Zeit des Zugriffes die Bildaufbereitung nicht richtig ablaufen, da für diese Zeit an den Datenausgängen des Speichers Werte stehen, die mit dem Inhalt der vom Prozessor adressierten Zelle übereinstimmen. Ist der Schreibstrahl des TV-Gerätes in diesem Moment an einer sichtbaren Stelle innerhalb des Zeichenfeldes, so erscheint dem Betrachter ein kurzer Fleck, der sich insbesondere bei einer hohen Einschreibrate des Prozessors störend bemerkbar macht.

Um diesen Flickereffekt zu vermeiden, kann zum Beispiel der Prozessor solange angehalten werden (z.B. mit WAIT), bis der Schreibstrahl in einer Austastlücke verschwunden ist (mit SYNC erkennbar), oder es wird der SYNC-Ausgang an einen IO-Port verbunden, der per Software von der CPU abgefragt wird.

Soll ein Zeichen in den Bildwiederholspeicher eingetragen werden, so wird das SYNC-Signal abgefragt. Falls das SYNC-Signal anzeigt, daß der Schreibstrahl nicht in der Austastlücke liegt, so wird solange gewartet, bis das SYNC-Signal wechselt. Dann werden die Zeichen eingeschrieben bis zu einem erneuten Wechsel des SYNC-Signals. Zur Abfrage kann auch das Blank-Signal verwendet werden, das das gültige Bildfeld anzeigt. *Abb. 1.3.1.3-3* zeigt die praktische Ausführung eines Bildwiederholspeichers. Die Multiplexer M1 bis M3 übernehmen die Umschaltung der Adressen. Sie werden mit dem Selekteingang gesteuert. Der Freigabeeingang ist mit Masse verbunden, da dauernd Zugriffe vorliegen. Der Selekteingang liegt im Normalfall auf einem Low-Pegel, dadurch sind die A-Eingänge auf den Ausgang durchgeschaltet und bestimmen die RAM-Adresse. Damit werden die Adressen des CRT-Controlers verwendet. Ein Zugriffswunsch liegt vor, wenn $\overline{\text{MRQ}}$ auf Low ist und die Adresse an V1 und V2 mit der eingestellten übereinstimmt. An dem Ausgang von V1 liegt ein High-Pegel, wenn die Adresse übereinstimmt und kein Refresh-Signal vorliegt. Die Vergleichsadresse wird mit J1 bis J5 eingestellt. Der Ausgang von N01 nimmt damit immer dann einen High-Pegel an, wenn ein Schreib- oder Lesezugriff vorliegt.

Der Datenbus muß ebenfalls entsprechend verschaltet werden. Bei einem Lesevorgang der CPU wird der Treiber B2 freigegeben, bei einem Schreibvorgang der Treiber B1. Bei einem Zugriff wird zunächst immer B1 freigegeben und erst, wenn das Signal $\overline{\text{RD}}$ erscheint, wird B2 freigegeben und B1 wieder gesperrt. Dadurch wird gewährleistet, daß bei einem Schreibzugriff die Daten vor dem R/$\overline{\text{W}}$-Signal gültig sind. Das R/$\overline{\text{W}}$-Signal wird durch Verknüpfung von $\overline{\text{WR}}$ und dem Signal für eine gültige Speicheradresse, das am Ausgang von N3 anliegt, gewonnen.

2. BAS-Mischer

Zur Steuerung eines TV-Monitors stehen vom CRT-Baustein im Prinzip zwei Signale zur Verfügung: Das SYNC-Signal und das Video-Signal. Diese beiden Informationen werden über eine Leitung übertragen, müssen also so gemischt werden, daß sie sich später vom Monitor wieder trennen lassen. Dazu ist eine analog arbeitende Schaltung nötig,

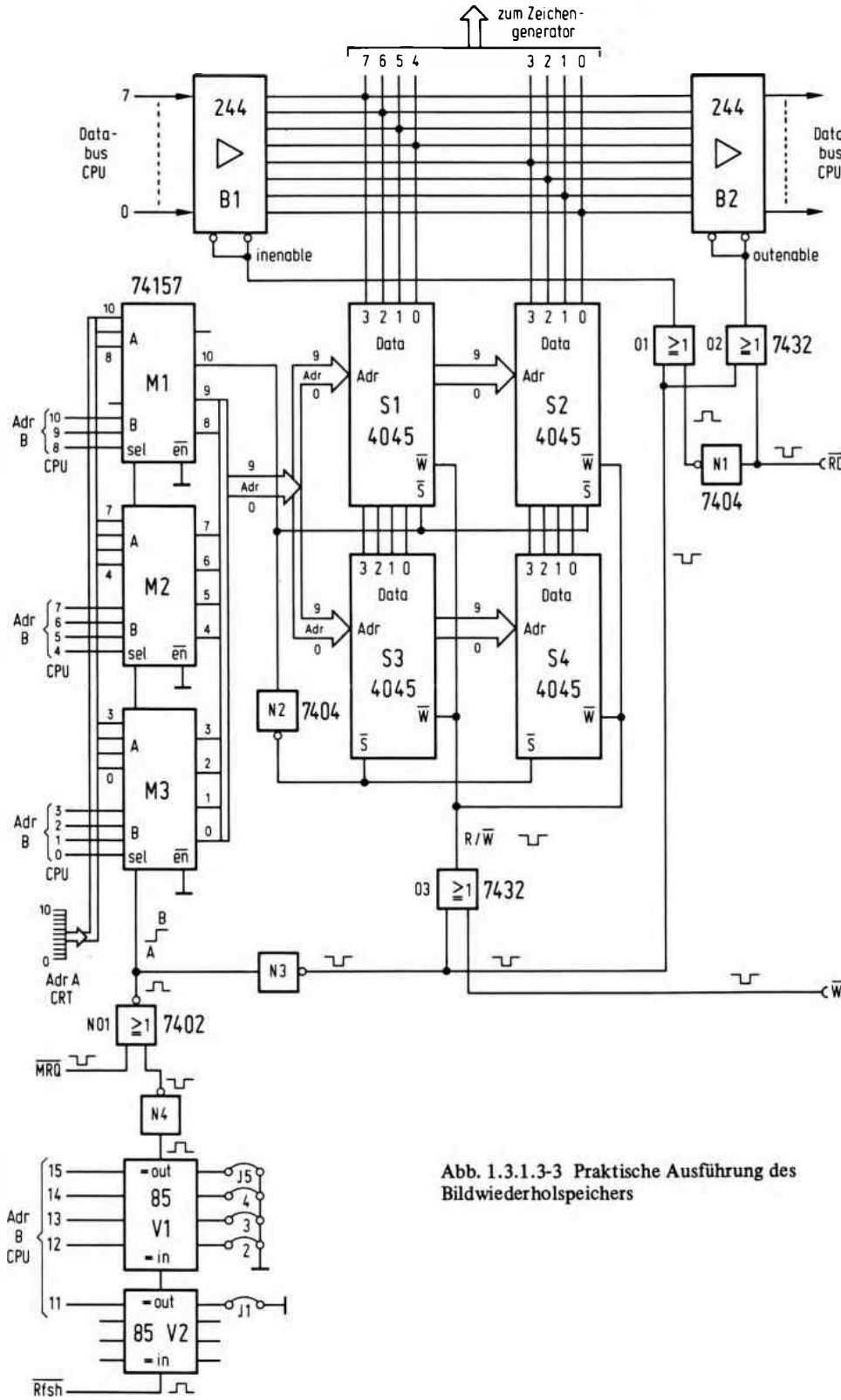


Abb. 1.3.1.3-3 Praktische Ausführung des Bildwiederholspeichers

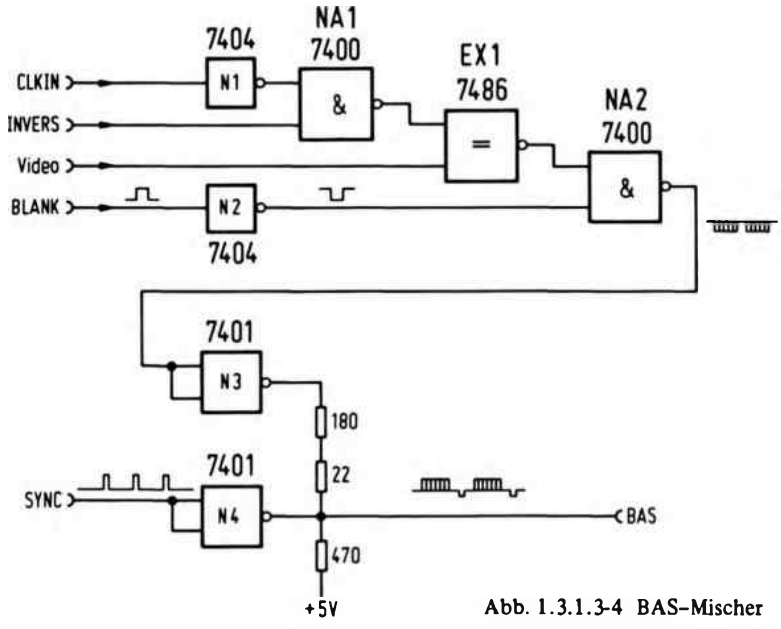


Abb. 1.3.1.3-4 BAS-Mischer

Abb. 1.3.1.3-4 zeigt die praktische Ausführung. Die Pegel des BAS-Signals sind genormt und werden von dieser Schaltung erzeugt. Mit N1, N2, NA1, NA2 und EX1 werden die einzelnen Teilsignale zu einem gesamten Videosignal gemischt, das an dem Eingang von N3 ansteht.

Der Eingang INVERS erlaubt die Darstellung von dunklen Zeichen auf hellem Grund und kann zur Auswahl zum Beispiel mit Bit 7 des Bildwiederholspeichers verbunden werden, wobei die übliche Schieberegisterschaltung dazwischen liegen muß.

Mit den Gattern N3 und N4, die einen offenen Kollektor besitzen, wird das BAS-Signal erzeugt.

3. HF-Generator

Wird kein Monitor verwendet, sondern ein gewöhnliches TV-Gerät, so ist es nötig, das BAS-Signal einem HF-Träger aufzumodulieren. Das Bild kann dann wie ein normales Fernsehprogramm empfangen werden (Vorschriften der Bundespost beachten!!!).

Abb. 1.3.1.3-5 zeigt die Schaltung. Beim Aufbau ist auf die Leitungsführung besonderer Wert zu legen. Die ganze Schaltung sollte auch in einem Metallgehäuse untergebracht werden, sowohl um Störungen benachbarter TV-Geräte zu vermeiden, als auch um Einstrahlungen durch den eigenen Computer zu vermeiden. Das gleiche gilt auch für den BAS-Mischer, der am besten zusammen mit dem HF-Generator abgeschirmt wird. Besonders kritisch an dieser Schaltung ist die Spule. Sollte der HF-Generator auf der falschen Frequenz schwingen, also nicht im normalen VHF-Bereich, so kann durch

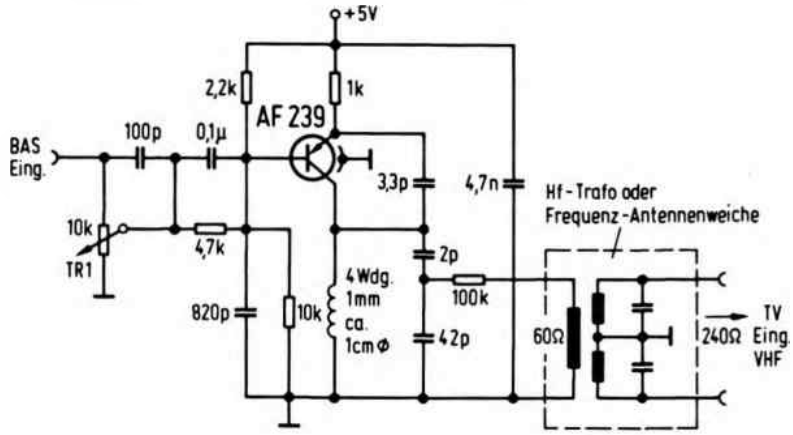


Abb. 1.3.1.3-5 HF-Generator

Wickeln einer neuen Spule ggf. eine Verbesserung erreicht werden. Auch ein sehr schlechtes Bild ist ein Anzeichen für eine falsche Frequenzlage. Mit dem Trimmer TR1 kann das Bild auf maximalen Kontrast eingestellt werden.

1.3.2 Drucker und deren Ansteuerung

Der Drucker stellt eines der wichtigsten Entwicklungshilfsmittel bei einem Computersystem dar. Es gibt eine Vielzahl von Druckern auf dem Markt, die leider sehr unterschiedliche Interfaceschaltungen nötig haben.

Es sollen hier drei verschiedene Arten von typischen Druckerschnittstellen besprochen werden, die den Leser in die Lage versetzen, einen Drucker an seinen Computer anzuschließen.

1.3.2.1 Paralleldrucker

Über Parallelleitungen kann die einfachste Druckerschnittstelle durchgeführt werden.

Abb. 1.3.2.1-1 zeigt eine typische Schaltung. Als Datenport des Computers wurde hier die Z80-PIO eingezeichnet. Port A ist als Ausgang programmiert und Port B als Eingang. Hier kann genausogut der 8255 Parallelbaustein verwendet werden.

Die Inverter I1 bis I9 dienen nur der Leitungspufferung. Von Port A gelangen sieben Datenleitungen über die Inverter an den Drucker, sie dienen der Übertragung der einzelnen Zeichen, sowie auch Steuerungsinformation, wie CR, LF etc. . Das Bit 7 von Port A wird als Strobesignal verwendet und ein kurzer Impuls auf dieser Leitung veranlaßt die Übernahme der Daten in den Drucker. Der Impuls wird dabei softwaremäßig erzeugt. Als Rückmeldung vom Drucker dient die Leitung BUSY, die angibt, ob der Drucker für den Empfang des nächsten Zeichens bereit ist. Dadurch wird gewährleistet, daß z.B. bei CR (Wagenrücklauf), der mehr Zeit benötigt, als der Ausdruck eines Zeichens, kein Zeichen verloren geht.

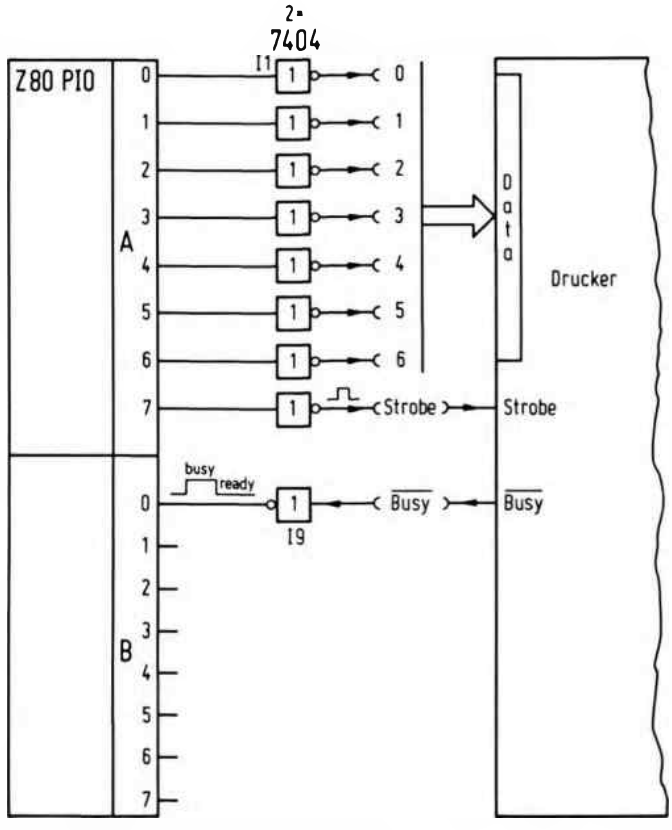


Abb. 1.3.2.1-1 Paralleldruckeranschluß

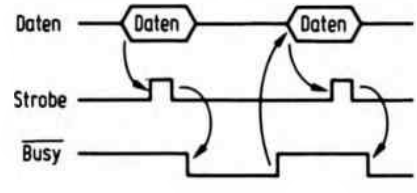


Abb. 1.3.2.1-2 Ablauf der Übertragung eines Zeichens

Abb. 1.3.2.1-2 zeigt den Impulsplan für den Ablauf der Übertragung eines Zeichens. Als erstes erfolgt das Anlegen der Daten auf den Datenbus. Es wird nun ein Strobeimpuls erzeugt, der die Daten in den Drucker einspeichert und den Ausdruck veranlaßt, bzw. bei Steuerzeichen die Ausführung. Nach der abfallenden Flanke des Strobesignals wird das BUSY-Signal ausgelöst, das solange auf Low bleibt, wie der Drucker mit den Daten beschäftigt ist. Bei manchen Druckern erscheint das BUSY-Signal nur bei Steuerbefehlen. Es muß dann im Programm eine entsprechende Entscheidung vorgesehen werden.

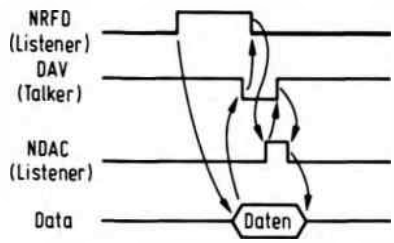


Abb. 1.3.2.1-3 Dreileiterhandshake (IEC)

```
000F'                   LOO1:
000F'   79              MOV A,C
0010'   FE0A            CPI 0AH
0012'   C8              RZ
0013'   F5              PUSH PSW
0014'                   ..LPMN:
0014'   DBFF            IN 0FFH
0016'   E601            ANI 1
0018'   28FA            JRZ ..LPMN
001A'   F1              POP PSW
001B'   2F              CMA             ;BINDER ROUTINE
001C'   F680            ORI 80H
001E'   D3FF            OUT 0FFH
0020'   E67F            ANI 7FH
0022'   D3FF            OUT 0FFH
0024'   F680            ORI 80H
0026'   D3FF            OUT 0FFH
0028'   2F              CMA             ;WERT MUSS ERHALTEN BLEIBEN
0029'   C9              RET
```

Abb. 1.3.2.1-4 Programm für Paralleldrucker

Abb. 1.3.2.1-3 zeigt ein ganz anderes Handshakeverfahren für eine Parallelschnittstelle. Es entspricht dem IEC-Bus-Dreileiterhandshake und erlaubt es auch, mehrere Geräte gleichzeitig an den BUS zu legen, die gemeinsam „mithören" können (LISTENERS) und dabei automatisch der langsamste Hörer die Übertragungsgeschwindigkeit auf dem Bus bestimmt.

Die drei Leitungen tragen die Bezeichnung NRDF (not ready for data), DAV (data available) und NDAC (not data accepted).

Im Grundzustand werden vom Sender (TALKER) Daten auf den Datenbus gelegt, dann wird die Leitung DAV auf LOW gelegt, um anzuzeigen, daß die Daten gültig sind. Darauf reagieren die Hörer mit einem LOW-Pegel auf der NRDF-Leitung. Sobald die Daten vom Hörer übernommen wurden, wird die NDAC-Leitung freigegeben (offener Ausgang). Das Signal NDAC wird erst dann auf High gehen, wenn auch der letzte Hörer die Daten übernommen hat. Danach wird DAV wieder auf High gelegt und die Information wird vom Datenbus weggenommen. Sobald NRDF wieder auf High geht, also alle Geräte mit der Verarbeitung fertig sind, kann sich das Handshake wiederholen.

Abb. 1.3.2.1-4 zeigt ein kurzes Druckprogramm, das für das erste Verfahren geeignet ist. Bei einem CALL auf dieses Programm wird das im Register C vorhandene Zeichen an

den Drucker übertragen. Die Routine ist in diesem Fall für den Binder Matrix Drucker BM 132 geschrieben. Als erstes wird abgefragt, ob das Steuerzeichen LF (0AH) übergeben wurde, falls ja, wird zum Hauptprogramm zurückgekehrt, sonst weiter ausgeführt. Dies ist nötig, da bei den meisten Programmen das Zeilenende mit CRLF abschließt, also Wagenrücklauf, Zeilenvorschub und bei diesem Drucker die Ausgabe von CR automatisch ein LF bewirkt. In der Schleife LPMN wird der BUSY-Ausgang des Druckers abgefragt und solange gewartet, bis die Leitung einen High-Pegel angenommen hat. Die Daten werden zunächst komplementiert, wegen der Inverter I1 bis I7 und mit ORI 80H wird Bit 7 auf 1 gelegt. Bei der nächsten Ausgabe wird durch den Befehl ANI 7FH Bit 7 auf 0 gelegt und anschließend wieder auf 1. Damit wird ein Strobesignal erzeugt. Die Daten werden anschließend erneut komplementiert und im Register A steht dann der ursprüngliche Wert.

1.3.2.2 Drucker mit Serienschnittstelle

Für die Druckinformation wird hier eine serielle Leitung verwendet. Für die Rückmeldung meist eine statische Leitung.

Die Daten werden mit Hilfe eines USARTs in eine serielle Information umgewandelt und an den Drucker geleitet. Über einen 1 Bit-Port wird die Rückmeldung vom Drucker empfangen.

Die Rückmeldung arbeitet meist genauso wie beim Parallelinterface und gibt an, ob das nächste Zeichen übertragen werden darf. *Abb. 1.3.2.2-1* zeigt das Prinzipschaltbild für den Anschluß eines solchen Druckers.

Abb. 1.3.2.2-2 zeigt ein Programm wie es für den Centronix Drucker 702 verwendet werden kann. Dabei ist in diesem Programm noch eine Steuerung des Protokollvorschubs vorgesehen. Wird die Zeile 66 erreicht, so erfolgt ein Vorschub zur Seitengrenze. Diese Maßnahme ist insbesondere bei amerikanischer Software ganz nützlich, da die Papierlänge dort i.a. 66 Zeilen beträgt. Der Vorschub wird hier mit dem Steuerzeichen „FORMFEED" erreicht, der Code ist 0CH. Über einen Schalter, der an Bit 4 von dem Port „PI0" angeschlossen ist, kann eingestellt werden, ob der Protokollvorschub gewünscht wird oder nicht. In der Speicherzelle „PRCOUNT" wird der Wert der aktuellen Zeile festgehalten. Diese Zelle ist vor Benutzung dieses Programms mit 0 vorzubesetzen.

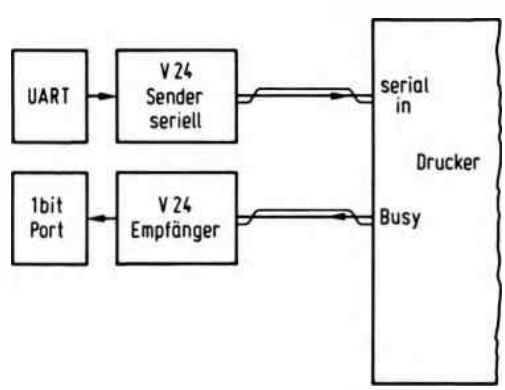


Abb. 1.3.2.2-1 Serienschnittstelle bei Druckern

```
LPRINTER:
IN PIO
ANI 0CH
JRNZ LPR1
MOV A,C          ;702 CENTRONIX PRINTER
CPI 0AH
JZ OPER
CALL 0F49DH      ;SERIALE PORT AUF SMB2
CPI 0DH          ;CR
JRZ WWAIT
CPI 9CH
JRZ WWAIT1
RET
WWAIT:
CALL BUSYL       ;IST PRINTER NOCH BUSY ?
MVI C,20H        ;WENN FERTIG DANN BLANK WEGEN
CALL 0F49DH      ;CR CR AUSGEBEN
RET
;
WWAIT1:
CALL BUSYL       ;NACH FORM FEED KEIN BLANK
XRA A            ;LOESCHEN FORMULARVORSCHUB
STA PRCOUNT
RET
;
OPER:
IN PIO
ANI 10H          ;AN ODER AUS?
RNZ              ;NORMAL PROT EIN
LDA PRCOUNT
INR A
STA PRCOUNT
CPI 42H          ;72 ZEILEN ALSO BEI 66 VORSCHUB
RNZ
XRA A
STA PRCOUNT
PUSH B
MVI C,0CH
CALL LPRINTER    ;VORSCHUB
POP B
RET
;
;
;
BUSYL:
PUSH B           ;WARTESYSTEM
PUSH PSW
MVI C,3          ;ETX AUSGEBEN
CALL 0F49DH
LP1:
IN PIO
RRC              ;BIT 0 DES PIOS ALS
JRC LP1          ;RUECKMELDUNG VERWENDEN
LP2:
IN PIO
RRC
JRNC LP2         ;1->0 0->1 ABWARTEN
POP PSW
POP B
RET              ;OK
;
LPR1:
JMP ERR
;
```

Abb. 1.3.2.2-2 Programm für den Drucker 702 (Centronix)

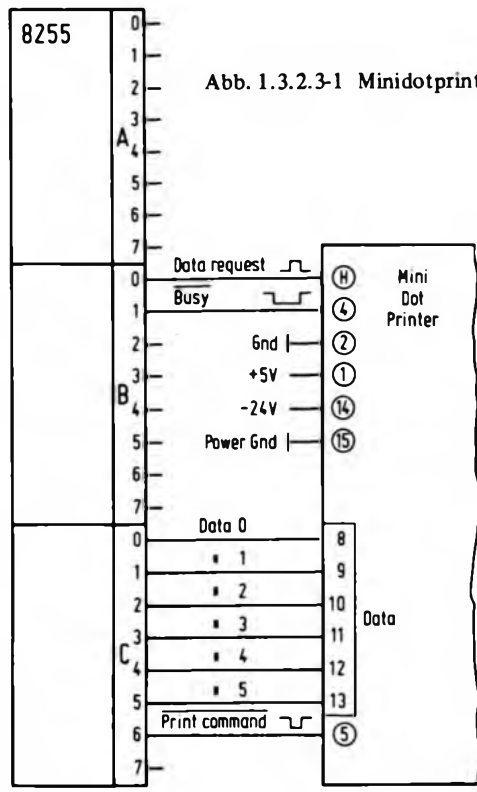


Abb. 1.3.2.3-1 Minidotprinter Anschluß über Parallelports

1.3.2.3 Drucker mit direkter Prozessorführung

Die bisher beschriebenen Interfaceverfahren verwenden einen Drucker, der oft selbst einen Prozessor oder eine Ablaufsteuerung verwendet. Es gibt aber auch eine Reihe von preiswerten Druckermodulen, die von einem externen Prozessor gesteuert werden müssen. Meist sind es Drucker mit schmalem Papier, aber auch große Druckermodule werden angeboten. Hier soll stellvertretend der „MINIDOTPRINTER" beschrieben werden, der mit schmalem Metallpapier arbeitet und maximal 32 Zeichen pro Zeile ausgeben kann. Er hat schon einen Zeichengenerator eingebaut, sowie die Treiberstufen für den Druckkopf, so daß nur noch die Daten synchron zum Druckvorgang vom Prozessor geliefert werden müssen.

Abb. 1.3.2.3-1 zeigt den Anschluß dieses Druckers über einen Parallelport des Typs 8255 an den Prozessor. Port C ist als Ausgang programmiert und Port B als Eingang. Der Drucker wird mit einem Puls auf der Leitung „PRINT COMMAND" gestartet und der Datentransfer wird mit den Leitungen „DATA REQUEST" und „BUSY" gesteuert.

Abb. 1.3.2.3-2 zeigt das dazugehörige Programm. Vor Benutzung der eigentlichen Druckroutine ist es nötig, den Programmteil INIT aufzurufen. Damit werden die verschiedenen Speicherzellen und der Parallelport initialisiert. Das Programm befindet

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 1
.MAIN. -


                        .PREL
                        .PHEX

                        ;**********************
                        ;* MINIDOT DRUCKER    *
                        ;* ROUTINE            *
                        ;* V790630            *
                        ;* R D K              *
                        ;**********************
                        ;
                        ; * DEFINITIONEN
00C7                    POCTRL=0C7H     ;CONTROL PORT PIA 8255
00C4                    POA=0C4H
00C5                    POB=0C5H
00C6                    POC=0C6H
                        ;
                        ;
                        ;
0000'                   LO:             ;GIBT EIN ZEICHEN
                                        ;AN DIE ROUTINE
                                        ;UND DRUCKT BEI CR
                                        ;ODER ZEILENUEBERLAUF
0000'   79              MOV A,C         ;ZEICHEN HOLEN
0001'   FE0D            CPI 0DH         ;IST ES CR
0003'   CA 0012'        JZ LOPRINT      ;JA DANN IN ROUTINE
0006'   FE20            CPI 20H         ;STEUERZEICHEN
0008'   D8              RC              ;IGNORIEREN
0009'   E65F            ANI 5FH         ;KLEIN IN GROSSBUCHSTABEN
000B'   4F              MOV C,A         ;WANDELN
000C'   0F              RRC
000D'   2F              CMA
000E'   E620            ANI 20H
0010'   B1              ORA C
0011'   4F              MOV C,A         ;WIEDER INS C REGISTER
                         ;
                        ;
                        ;
0012'                   LOPRINT:        ;EIGENTLICHE DRUCKERROUTINE
0012'   C5              PUSH B
0013'   D5              PUSH D
0014'   E5              PUSH H          ;REGISTER RETTEN
0015'   79              MOV A,C         ;ZEICHEN HOLEN
0016'   FE0D            CPI 0DH         ;CR ?
0018'   2815            JRZ PRLINE      ;DRUCKEN !
001A'   2A 009A'        LHLD STO1       ;POINTER HOLEN
001D'   71              MOV M,C         ;ZEICHEN ABSPEICHERN
001E'   23              INX H           ;POINTER UM EINS ERHOEHEN
001F'   22 009A'        SHLD STO1       ;WIEDER IN ZWISCHENSPEICHER
0022'   3A 009C'        LDA STO2        ;LAENGENZAEHLER
0025'   3D              DCR A
0026'   281D            JRZ PRLIBL      ;DRUCKE MIT BLANK
0028'   32 009C'        STA STO2        ;NEUER ZAEHLERSTAND
002B'                   POPRET:
002B'   E1              POP H           ;ALLE REGISTER ZURUECK
002C'   D1              POP D
```

Abb. 1.3.2.3-2 Steuerprogramm für den Minidotprinter

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 2
.MAIN. -


002D'   C1              POP B
002E'   C9              RET
                        ;
002F'                   PRLINE:         ;DRUCKE ZEILE
002F'   3EE0            MVI A,0E0H      ;ENDE ZEICHEN
0031'   2A 009A'        LHLD STO1       ;POINTER
0034'   77              MOV M,A         ;ZEICHEN ABLEGEN
0035'   21 009D'        LXI H,BUFFA     ;ZEICHENBUFFER
0038'   22 009A'        SHLD STO1       ;POINTER AUF ANFANG
003B'   3E20            MVI A,20H       ;ZEILENLAENGE
003D'   32 009C'        STA STO2        ;ABSPEICHERN
0040'   CD 0069'        CALL LINE       ;DRUCKEN
0043'   18E6            JMPR POPRET
                        ;
0045'                   PRLIBL:         ;DRUCKEN MIT BLANK
0045'   36E0            MVI M,0E0H      ;ENDE ZEICHEN
0047'   CD 0069'        CALL LINE       ;DRUCKEN
004A'   21 009D'        LXI H,BUFFA     ;ZEICHENBUFFER
004D'   3E20            MVI A,20H       ;ZEILENLAENGE
004F'   32 009C'        STA STO2
0052'   3620            MVI M,20H       ;IN NAECHSTE ZEILE ALS
                                        ;ERSTES BLANK
0054'   23              INX H           ;POINTER + 1
0055'   22 009A'        SHLD STO1       ;NEUER POINTER
0058'   18D1            JMPR POPRET
                        ;
                        ;
005A'                   PRINT:          ;ZEICHEN DRUCKEN
005A'   F5              PUSH PSW
005B'   D3C6            OUT POC
005D'                   LP1:
005D'   DBC5            IN POB
005F'   0F              RRC             ;WARTEN 1->0
0060'   38FB            JRC LP1
0062'                   LP2:
0062'   DBC5            IN POB
0064'   0F              RRC             ;WARTEN 0->1
0065'   30FB            JRNC LP2
0067'   F1              POP PSW
0068'   C9              RET
                        ;
0069'                   LINE:
0069'   3E00            MVI A,0         ;STARTEN DRUCKER
006B'   D3C6            OUT POC
006D'   21 009D'        LXI H,BUFFA
0070'                   LP11:
0070'   7E              MOV A,M         ;ZEICHEN AUS BUFFER HOLEN
0071'   23              INX H
0072'   CD 005A'        CALL PRINT      ;AUSGEBEN
0075'   FEE0            CPI 0E0H        ;ENDE ZEICHEN
0077'   20F7            JRNZ LP11
0079'                   ENDL:
0079'   DBC5            IN POB          ;WARTEN BIS DRUCKER FERTIG
007B'   E602            ANI 2           ;BUSY FLAG
007D'   28FA            JRZ ENDL
```

zu Abb. 1.3.2.3-2

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 3
.MAIN. -


007F'                   LP21:
007F'   DBC5            IN P0B          ;LETZER DATA REQ
0081'   E601            ANI 1
0083'   20FA            JRNZ LP21
0085'   C9              RET
                        ;
                        ;
0086'                   INIT:           ;MUSS EINMAL AM ANFANG
                                        ;AUFGERUFEN WERDEN
0086'   3E92            MVI A,92H       ;PIA INITIALISIEREN
0088'   D3C7            OUT P0CTRL
008A'   3EFF            MVI A,0FFH      ;AUSGAENGE AUF HIGH
008C'   D3C6            OUT P0C
008E'   21 009D'        LXI H,BUFFA     ;POINTER DEFINIEREN
0091'   22 009A'        SHLD ST01
0094'   3E20            MVI A,20H       ;ZEILENLAENGE
0096'   32 009C'        STA ST02
0099'   C9              RET
                        ;

                        ;
                        ;
                        ; *SPEICHERZELLEN *
009A'   009D'           ST01:   .WORD BUFFA
009C'   20              ST02:   .BYTE 20H
009D'                   BUFFA:  .BLKB 21H
                        ;
                        .END
```

zu Abb. 1.3.2.3-2

sich auf der Adresse 86H. Das Druckprogramm startet bei der Marke L0 und im C-Register wird das zu druckende Zeichen übergeben.

Der Druckvorgang wird gestartet, nachdem ein CR gegeben wurde, oder der Druckpuffer mit mehr als 32 Zeichen gefüllt wurde. Im letzteren Fall wird in der nächsten Zeile ein führendes Blank ausgegeben.

1.3.3 Kassettengerät als Datenspeicher

Da nach dem Abschalten der Versorgungsspannung der Informationsinhalt von statischen Speichern verloren geht, ist es nötig, Daten und Programme auf einem nicht flüchtigen Speicher festzuhalten.

Hier soll ein preiswertes Interface für einen Kassettenrecorder beschrieben werden, das diesen in einen Datenspeicher mit hoher Kapazität (100K Bytes) verwandelt. Dabei wird hier eine Schaltung verwendet, die sich durch hohe Zuverlässigkeit und durch eine hohe Datenrate (1200 Baud) auszeichnet. Die hohe Sicherheit wird durch eine Phasenmodulation erreicht. Durch dieses Verfahren ist eine hohe Baudrate möglich, ohne daß die Aufzeichnungsfrequenz höher wird, im Gegensatz zum Beispiel zum Frequenzshiftverfahren, bei dem zwischen zwei verschiedenen Frequenzen umgeschaltet wird, die aber ca. 5fach höher liegen, als die Bitwechselzahl der Datenübertragung.

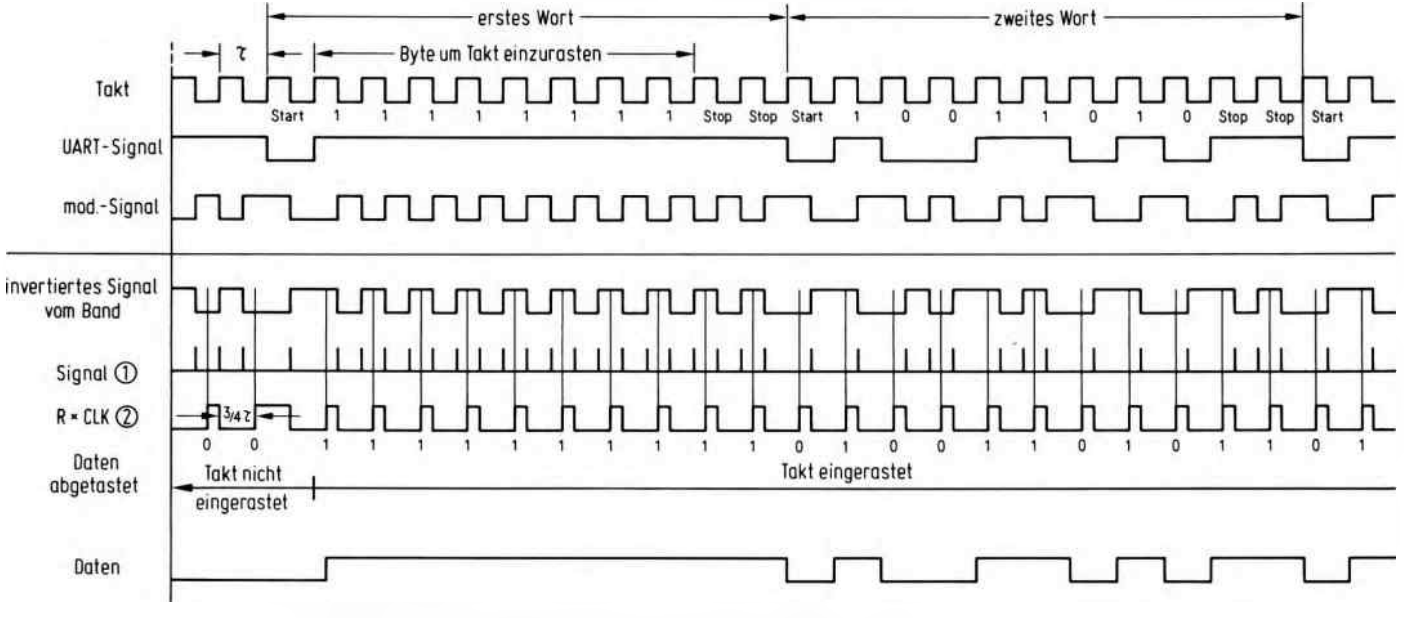


Abb. 1.3.3-1 Impulsplan zum Phase-encoding Verfahren (PE)

Abb. 1.3.3-1 zeigt das Impulsdiagramm zum PE-Verfahren. Die Daten werden in ein Seriensignal umgewandelt, wobei dies mit einem UART geschieht. *Abb. 1.3.3-2* zeigt die dazugehörige Schaltung.

Der UART wird für die Taktart 1:1 programmiert. Der Takt, der über das Flipflop FF1 symmetriert wird, gelangt an den TxCLK-Eingang des UART und an den Eingang des Exclusiv-ODER-Gatters EX3. Dort wird er mit dem Datensignal des UART-Ausgangs verknüpft, und es entsteht das phasenmodulierte Signal. Es wird dem Kassettenrecorder über einen Spannungsteiler zugeführt und ist gleich spannungsfrei, bezogen auf den virtuellen Massepunkt, mit dem der Kassettenrecorder verbunden ist.

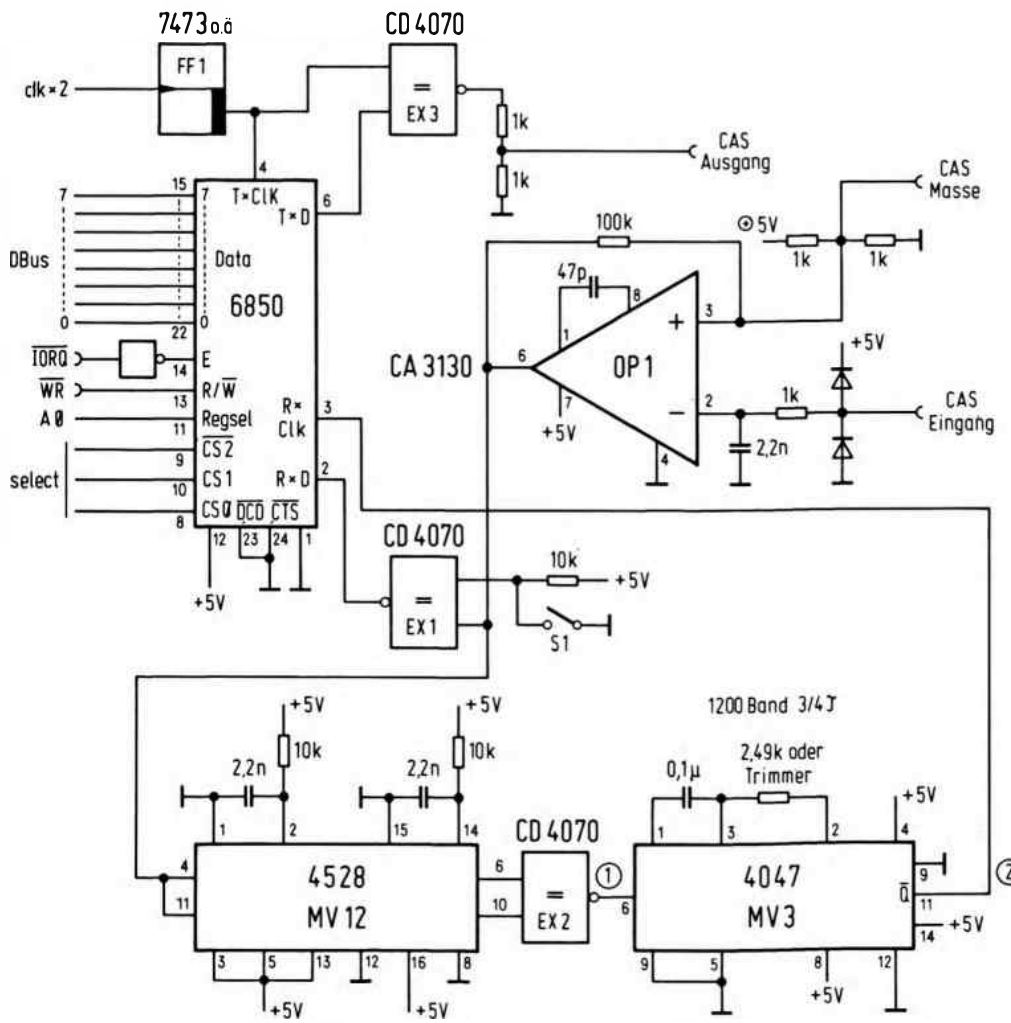


Abb. 1.3.3-2 Schaltung des Kassetteninterface mit PE

Bei einer Baudrate von 1200 Baud beträgt die Bitwechselfrequenz 1,2 kHz. Dadurch ist es möglich, auch Rekorder mit einer relativ niederen Grenzfrequenz zu verwenden, ein Versuch wurde mit einem Minidiktiergerät mit 2,3 cm/s erfolgreich durchgeführt.

Die Demodulation ist etwas aufwendiger. Das Signal wird vom Ausgang des Kassettenrecorders (Radiobuchse oder Kopfhörerbuchse mit Spannungsteiler) an einen Operationsverstärker geleitet, der als Nulldurchgangsdetektor geschaltet ist.

Da der UART im Mode 1:1 arbeitet, ist es nötig, einen synchronen Takt zu erzeugen. Dies geschieht mit Hilfe dreier Monoflops, von denen nur einer eine kritische Zeitkonstante besitzt.

Das Ausgangssignal des Operationsverstärkers wird an ein Exclusiv-ODER-Gatter EX1 geführt und dessen Ausgang gelangt direkt an den Eingang RxD. Dies ist nur bei dem 6850 möglich, da beim 8251 ein statisches Eingangssignal verlangt wird. Der andere Eingang des Gatters EX1 ist an einen Schalter geführt, mit dem eine im Kassettenrecorder verursachte Polaritätsumkehr rückgängig gemacht werden kann. Das Ausgangssignal des OPs gelangt ferner an die Eingänge zweier Monoflops, von denen eines auf die positive, das andere auf die negative Flanke reagiert. Am Ausgang von EX2 entsteht der im Impulsplan mit Signal 1 bezeichnete Takt. Er gibt an, wann ein Bitwechsel stattfindet. Signal 1 wird nun einem weiteren Monoflop zugeführt, das an seinem Ausgang den synchronen Takt liefert. Dazu wird die Zeitkonstante des Monoflops auf 3/4 der Periodendauer des Baudratetaktes eingestellt. Nach dem Einschalten des Kassettenrecorders ist der Takt im allgemeinen nicht eingerastet, und es können invertierte Daten gelesen werden. Die Polarität wird aber nach dem ersten Bitwechsel des Datensignals automatisch festgestellt und der Takt rastet ein. Dazu muß vor jeder Übertragung eines Datenblockes ein zusätzliches Zeichen am Anfang mit übertragen werden, um die Synchronisierung ohne Datenverlust zu erreichen, im Impulsdiagramm ist der Einrastvorgang erkennbar.

Inbetriebnahme des Interface:

Die Zeitkonstante des Monoflops MV3 wird als erstes eingestellt. Dazu wird eine Impulsfolge an den Eingang des Monoflops gelegt und der Ausgang mit einem Oszilloskop oder mit einem entsprechenden Digitalmeßgerät beobachtet. Die Einstellung sollte aber nicht dadurch vorgenommen werden, daß der Trimmer solange verdreht wird, bis die Daten korrekt erscheinen, da dadurch die optimale Einstellung wegen des breiten Bereichs nicht gefunden werden kann. Als nächstes wird ein Takt von 2,4 kHz an den Eingang CLK x 2 geschaltet, der an das Flipflop FF1 führt.

1200 Baud stellt die untere Baudrate dar, bei der dieses Verfahren arbeitet, bei einer geringeren Baudrate ergeben sich Schwierigkeiten durch die untere Grenzfrequenz des Kassettengerätes. Hingegen ist die maximale Rate 4800 Baud. Zum weiteren Test wird ein Programm geschrieben, das es erlaubt, ein Zeichen von der Tastatur einzugeben und dieses Zeichen über das Interface auf dem Rekoder abzuspeichern.

Nach Starten dieses Programms werden eine Reihe von Zeichen über die Tastatur eingegeben und aufgezeichnet. Dann wird ein Programm gestartet, das genau umgekehrt arbeitet. Es liest ein Zeichen vom Interface ein und gibt es auf dem Terminal aus. Es müßten nun die eingegebenen Zeichen auf dem Bildschirm erscheinen. Ist dies nicht der Fall, so kann einmal mit dem Oszilloskop der Ausgang des Operationsverstärkers betrachtet werden, und falls eine Übersteuerung vorliegt, so muß ein zusätzlicher Ein-

gangsspannungsteiler eingebaut werden. Als nächstes wird die Polarität mit S1 vertauscht und die Daten müßten nun erscheinen. Falls immer noch nicht, dann muß eine Überprüfung der Schaltung anhand des Impulsdiagramms vorgenommen werden.

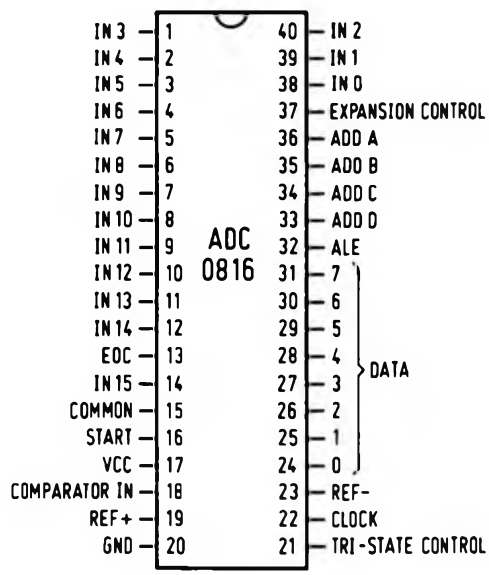


Abb. 1.3.4-1 PIN-Belegung des A/D-Umsetzers ADC 0816

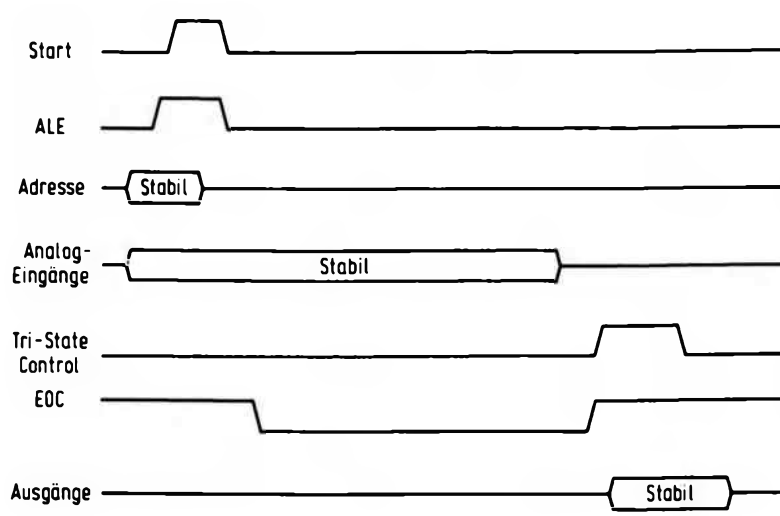


Abb. 1.3.4-2 Ablauf eines Meßvorgangs

1.3.4 A/D-Umsetzer

Sehr interessant ist der Anschluß von analogen Signalquellen. Hierzu ist es nötig, die analoge Information in ein digitales Signal umzuwandeln.

Dazu dienen die A/D-Umsetzer. Es sind heute eine Vielzahl von Bausteinen auf dem Markt, die den direkten Anschluß an einen Mikrocomputer ermöglichen. Hier soll nur ein Beispiel besprochen werden, der Baustein ADC 0816, der von NATIONAL SEMICONDUCTOR und MOSTEK geliefert wird. Dieser Baustein enthält einen 8-Bit-A/D-Umsetzer und einen 16-Kanal-Analogmultiplexer. Es ist dann möglich, 16 analoge Signalquellen mit diesem Baustein zu verbinden. *Abb. 1.3.4-1* zeigt die PIN-Belegung dieses Bausteins. Er benötigt zum Betrieb nur eine 5-V-Versorgungsspannung, sowie zwei Referenzspannungen REF+ und REF-. Als Referenzspannungen können aber auch + 5 V und Masse verwendet werden. Der Datenausgang ist TTL kompatibel und mit TRI-State-Treibern ausgerüstet, so daß er direkt an einen Prozessorbus angeschlossen werden kann.

Abb. 1.3.4-2 zeigt den Ablauf eines Meßvorgangs. Die Kanaladresse wird an die vier Adreßleitungen gelegt und mit dem Signal ALE in den internen Speicher übernommen. Mit START wird der Meßvorgang eingeleitet. Während der Meßdauer muß der analoge Eingang einen stabilen Wert aufweisen. Das Signal EOC geht auf Low, um anzuzeigen, daß die Messung noch nicht beendet ist. Nachdem das Signal EOC wieder den High-Pegel angenommen hat, können die Daten durch Öffnen der Tri-State-Treiber mit dem Signal TRI-STATE CONTROL vom Prozessor gelesen werden.

Ein Meßvorgang dauert nur ca. 100 ys, je nach Takt, da eine sukzessive Approximation für die Ermittlung des Meßwertes verwendet wird.

Abb. 1.3.4-3 zeigt die praktisch ausgeführte Schaltung. Der A/D-Umsetzer belegt 16 Adressen von 0B0H bis 0BFH. Die Adresse wird mit dem Gatter N2 bestimmt. Zur Abfrage der Rückmeldeleitung EOC wird ein 1 Bit Port benötigt, der mit dem Treiber B2 realisiert ist. Die Adresse des 1 Bit Ports ist mit N1 auf 0C8H eingestellt. Als Takt für den Umsetzer wird eine Frequenz von 600 kHz verwendet, die aus einem 1,2-MHz-Takt durch Teilung mit FF1 gewonnen wird. Dieser Takt bestimmt die Meßdauer. Die beiden Referenzspannungen REF+ und REF- werden mit Hilfe des Spannungsteilers R1, TR1, R2, TR2, R3 gewonnen. Die damit erreichte Genauigkeit und Stabilität ist in den meisten Fällen ausreichend.

Mit dem Trimmer TR1 kann der obere Bereich von ca. 4 bis 5 V eingestellt werden und mit TR2 der Bereich von 0 bis 1 V. Günstig ist eine Wahl von 4 V für den oberen und 1 V für den unteren Bereich, um zu gewährleisten, daß nur der lineare Teil des Operationsverstärkers LM 3900 verwendet wird. Die Operationsverstärker können dann auch nur mit 5 V versorgt werden.

Abb. 1.3.4-4 zeigt den Ablauf eines Meßvorgangs in Flußdiagrammform. Als erstes wird durch Ausgabe eines beliebigen Datenwertes auf die dem gewünschten Kanal zugeordnete Portadresse der Meßvorgang gestartet. Dann wird auf den Wechsel High Low und Low High des EOC-Ausgangs gewartet, und anschließend kann der Meßwert eingelesen werden. *Abb. 1.3.4-5* zeigt die praktische Ausführung eines Assemblerprogramms. Es wird als Unterprogramm aufgerufen, und im Register C wird die Kanaladresse (0 bis FH) übergeben. Das Ergebnis des Meßvorgangs ist dann im Register A verfügbar.

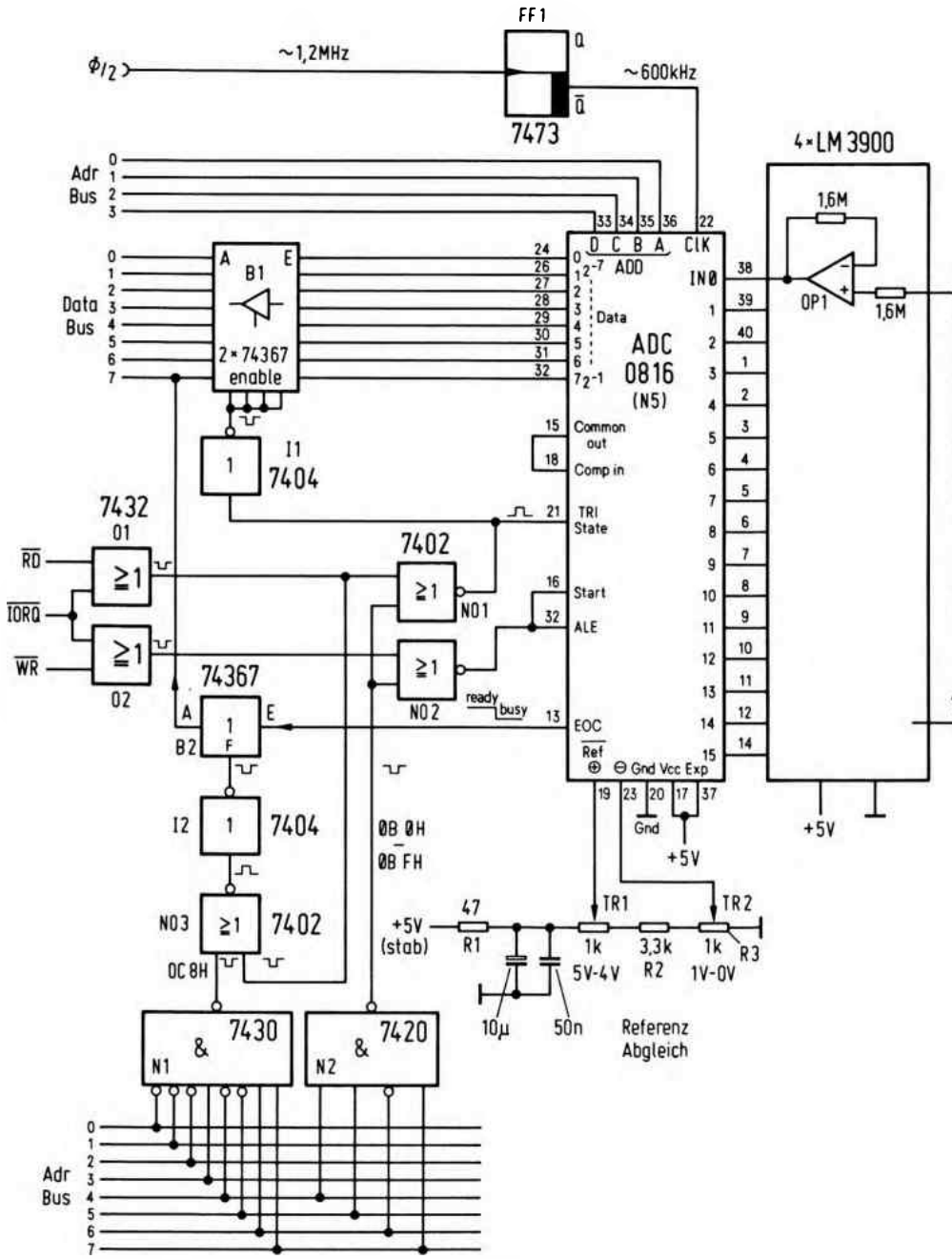


Abb. 1.3.4-3 Anschluß des ADC 0816 an den Z80

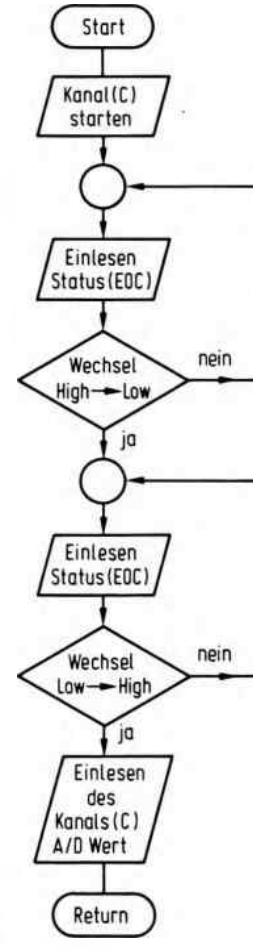


Abb. 1.3.4-4 Flußdiagramm eines Meßablaufes

Abb. 1.3.4-5 Assemblerprogramm zur Meßdatenerfassung

```
                        ;**************************
                        ;* A/D UMSETZER ROUTINE   *
                        ;* RDK 790318             *
                        ;**************************
0000'   79              AD:     MOV A,C         ;IN C IST KANAL (0...F)
0001'   E60F                   ANI 0FH         ;NUR 16 KANAELE
0003'   F6B0                   ORI 0B0H        ;IO PORT BERECHNEN
0005'   4F                     MOV C,A         ;FUER AUSGABE BEFEHL
0006'   ED79                   OUTP A          ;DUMMY NACH (C) AUSGEBEN
0008'   DBC8            LP:     IN 0C8H         ;WARTEN
000A'   E680                    ANI 80H         ;HIGH LOW WECHSEL
000C'   20FA                   JRNZ LP         ;ABWARTEN
000E'   DBC8            LP1:    IN 0C8H         ;NUN AUF LOW HIGH
0010'   E680                   ANI 80H         ;WECHSEL WARTEN
0012'   28FA                   JRZ LP1
0014'   ED78                   INP A           ;UEBER (C) WERT HOLEN
0016'   C9                     RET             ;FERTIG
                                .END            ;ENDE DES PROGRAMMS

++++++ SYMBOL TABLE +++++

AD      0000'           LP      0008'           LP1     000E'
```

Abb. 1.3.4-6 zeigt ein BASIC-Programm mit der Aufgabe, alle 16 Kanäle zu protokollieren. Auf Zeile 1000 startet das Unterprogramm für die Umwandlung. Es ist nicht nötig auf das EOC-Signal zu warten, da das BASIC-Programm relativ langsam arbeitet und der zeitliche Abstand in der Ausführung zweier Zeilen größer als 100 ys ist.
Im unteren Bildteil ist das Ergebnis der Protokollierung abgebildet, nur drei der 16 Kanäle sind mit analogen Quellen belegt.

Abb. 1.3.4-7 zeigt die Schaltung einer solchen Analogquelle. Es handelt sich dabei um einen „Joystick". Dies ist ein Kreuzknüppelpotentiometer wie es von Flugzeugfernsteuerungen her bekannt ist. Es besteht aus zwei getrennt einstellbaren Potentiometern.

```
10 REM AD Umsetzer Data aquisitation program
20 REM RDK 790318
30 FOR I=0 TO 15
40 LPRINT USING 230;I;
50 NEXT I                    'Ueberschrift
60 LPRINT
70 LPRINT
100 DIM W(16)    '16 Kanaele
110 FOR J=1 TO 10            '10 mal Ueberwachung
120 FOR I=0 TO 15            'Werte holen
130 C=&B0 OR I               'Port Adresse berechnen
140 GOSUB 1000
150 W(I)=A                   'Wert uebergeben
160 NEXT I
170 FOR I=0 TO 15            'Werte ausgeben
180 LPRINT USING 230;W(I);
190 NEXT I
200 LPRINT                   'Zeilenvorschub
210 NEXT J
220 STOP
230 ' ###
1000 REM Unterprogrammzur Umwandlung
1010 REM in der Variablen C wird der
1020 REM Kanal eingegeben in A erhaelt
1030 REM man den AD Wert.
1040 OUT C,0     'A/D Umsetzer starten
1050 A=INP(C)    'Wert holen
1060 RETURN      'von UPR zurueck
```

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

 77   0  11   0   0   0   0   0   0   0   0   0   0   0   0   0
 93   0   0  26   0   0   0   0   0   0   0   0   0   0   0   0
125   0   9   0   0   0   0   0   0   0   0   0   0   0   0   0
138   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0
157   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
140   0   9   0   0   0   0   0   0   0   0   0   0   0   0   0
122   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
103   0   0  21   0   0   0   0   0   0   0   0   0   0   0   0
 78   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 62   0   0  26   0   0   0   0   0   0   0   0   0   0   0   0
```

Abb. 1.3.4-6 BASIC-Programm zur Datenerfassung

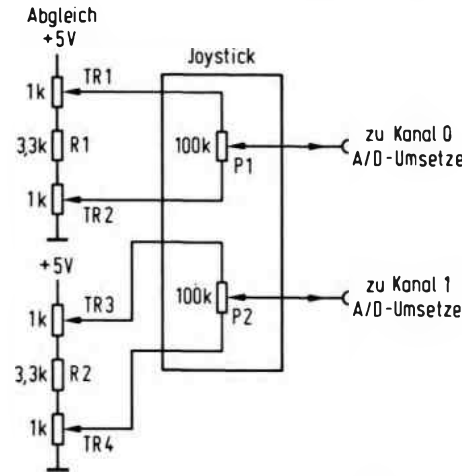


Abb. 1.3.4-7 Schaltung eines „joysticks"

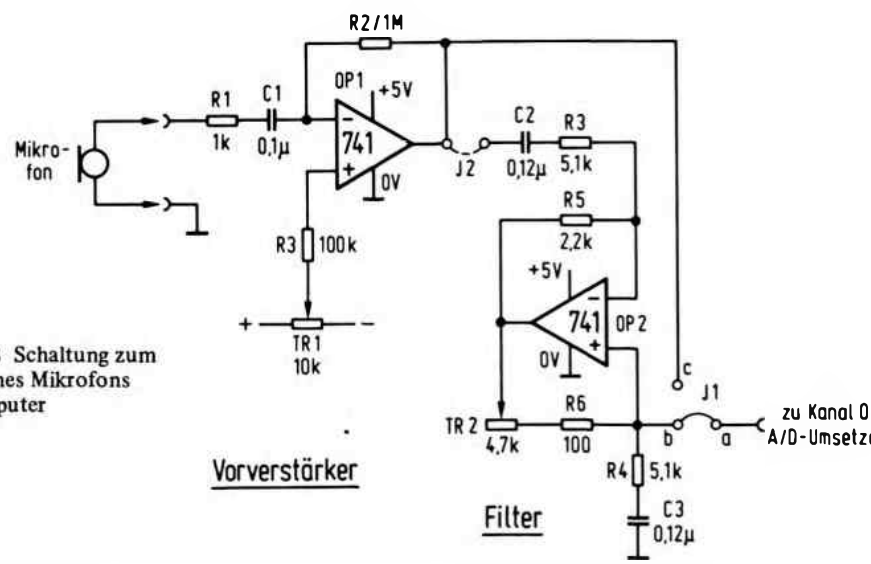


Abb. 1.3.4-8 Schaltung zum Anschluß eines Mikrofons an den Computer

Mit den Trimmern TR1 bis TR4 können die Ausgangsspannungsbereiche der beiden Potis P1 und P2 so eingestellt werden, daß sie den ganzen Eingangsspannungsbereich des A/D-Umsetzers erfassen.

Die Anwendungen des Joysticks sind sehr verschiedenseitig. Es können damit anspruchsvolle Spiele realisiert werden, aber auch die Eingabe von Kurvenzügen für Funktionsgeneratoren ist denkbar, sowie die analoge Einstellung von digitalen Parametern in einem Programm, wie z.B. kritische Warteschleifen. *Abb. 1.3.4-8* zeigt eine ganz andere Analogquelle. Mit dieser Schaltung ist es zum Beispiel möglich, Sprache zu digitalisieren. Dies geschieht über ein Mikrofon, dessen Ausgang an den Operationsverstärker

OP1 geführt wird. Danach könnte das Signal schon direkt einem A/D-Umsetzer zugeführt werden, doch es ist sehr vorteilhaft, wenn das Signal zuvor noch über eine Filterschaltung geführt wird. Das Filter ist mit dem Operationsverstärker OP2 realisiert, das im wesentlichen die Höhen beschneiden soll. Dadurch ist gewährleistet, daß sich das Abtasttheorem erfüllen läßt und eine störungsfreie Aufzeichnung gewährleistet wird. Bei dem Filter handelt es sich um einen Bandpaß, dessen Mittenfrequenz durch R3 und C2 bestimmt ist. Dabei müssen R4 und C3 die gleichen Werte besitzen. Die Güte des Filters ist mit TR2 einstellbar und sollte hier nicht zu hoch gewählt werden.
Mit dieser Anordnung können nun sehr vielseitige Experimente durchgeführt werden. Hier nur einige Anregungen.

Digitale Sprachaufzeichnung:

Aufgabe ist es, den Hauptspeicher des Rechners für eine Sprachaufzeichnung zu verwenden, ähnlich zu einem Tonbandgerät. Probleme ergeben sich dabei bezüglich der begrenzten Speicherkapazität. Für eine gute Qualität ist für eine Sekunde Sprechdauer 50K Bytes anzusetzen. Eine Verbesserung erreicht man durch Verwendung von Sprachkompression. Dabei besteht die Möglichkeit, mit weniger als 8 Bits aufzuzeichnen. Im Extremfall nur ein Bit. Die dabei aufgezeichnete Sprache bleibt durchaus noch verständlich, doch ergeben sich Probleme wegen des geringen Dynamikbereichs, so daß nur mit einer bestimmten Lautstärke bei der Aufzeichnung die Sprache verständlich bleibt. Eine andere Möglichkeit liegt darin, nur Impulspakete aufzuzeichnen, indem zum Beispiel für die Länge eines Schwingungszuges eines Vokals aufgezeichnet und dann eine längere Pause eingelegt wird, zum Beispiel mit einem Tastverhältnis 1:5. Bei der Wiedergabe über den D/A-Umsetzer werden die fehlenden Analogdaten durch Wiederholung des Schwingungszuges, der in der Aufzeichnungsphase festgehalten wurde, ergänzt. Die so wiedergegebene Sprache ist verständlich, klingt aber sehr metallisch.

Digitale Spracherkennung:

Es wurden hier schon viele Versuche unternommen. Es handelt sich aber um ein sehr schwieriges Vorhaben, da die Analyse der menschlichen Sprache sehr komplizierte Ähnlichkeitsverfahren benötigt. Es besteht zum Beispiel die Möglichkeit, die Sprache mit einer Fourieranalyse in die Frequenzanteile aufzuspalten und dann mit einer Korrelationsanalyse Vergleiche mit vorhandenen Sprachmustern durchzuführen. Probleme ergeben sich bei der Erkennung von einzelnen Worten innerhalb eines flüssig gesprochenen Textes, doch die Analyse einzelner Worte ist leichter möglich, wenn sie einzeln gesprochen werden.

Ein anderes Problem stellt die Übertragbarkeit dar. Der vorhandene Vergleichssatz wird z.B. von einer Versuchsperson erstellt. Dem Rechner wird es dann sehr schwer fallen, die Sprache einer anderen Person zu verstehen. Möglichkeiten, um diesem Problem zu begegnen, sind evtl. eine Lernphase, in der der Rechner auf verschiedene Personen trainiert wird, und daraus ein gemeinsames Erkennungsmuster entwickelt.

Digitale Sprechererkennung:

Im Prinzip ähnlich schwierig ist es, Personenidentifizierung anhand eines Probetextes. Dabei kann zum Beispiel die Nulldurchgangsanzahl gemessen werden, sowie verschiedene Spektralanteile, die für einen Sprecher sehr typische Werte annehmen können.

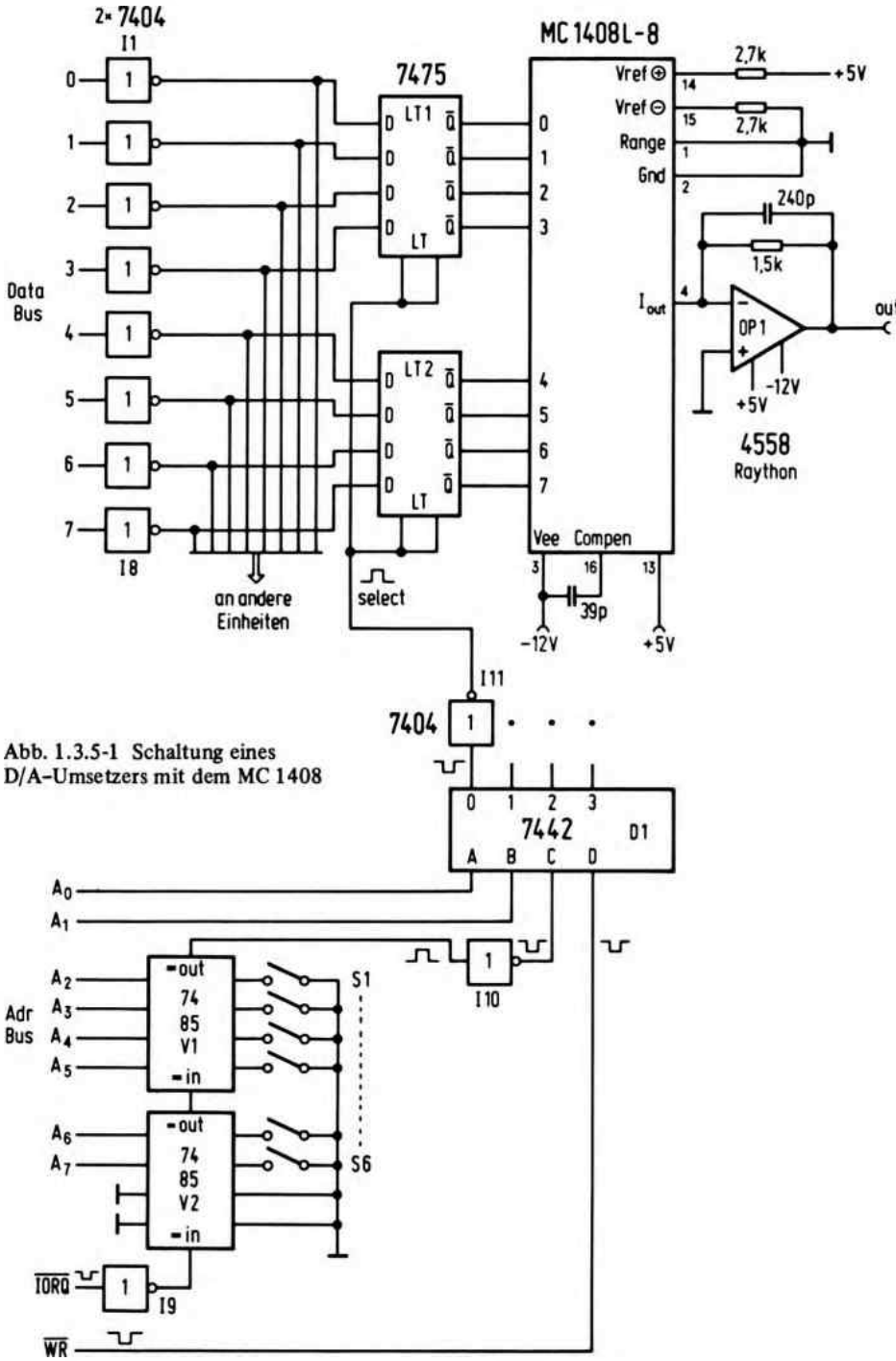


Abb. 1.3.5-1 Schaltung eines D/A-Umsetzers mit dem MC 1408

1.3.5 D/A-Umsetzer

D/A-Umsetzer sind einfacher zu verwirklichen als A/D-Umsetzer. Es gibt auch dafür eine Vielzahl an integrierten Bausteinen, von denen ein 8-Bit-Umsetzer als Beispiel gewählt wurde. *Abb. 1.3.5-1* zeigt die dazugehörige Schaltung. Es werden vier Adressen belegt, deren Bereich mit den Schaltern S1 bis S6 eingestellt werden kann. Die Daten gelangen über die Inverter I1 bis I8, die den Bus nur schwach belasten, an einen internen Bus. Die Eingänge der beiden Latches LT1 und LT2 sind mit dem internen Bus verbunden. Die Latches halten die Information, die in einen analogen Wert gewandelt werden soll, fest. Dazu sind die Q-Ausgänge der Latches an einen D/A-Umsetzer geführt.
Die Schaltung ist für den Anschluß von vier D/A-Umsetzern ausgelegt und die Dekodierung erfolgt mit dem Dekoder D1, dessen Ausgänge über Inverter an den Strobeeingang der Latches gelangt. Es erscheint immer dann ein Impuls, wenn ein Schreibvorgang auf einen IO-Port stattfindet und die richtige Adresse anliegt.

Als D/A-Umsetzer wurde der preiswerte Typ MC 1408 L-8 gewählt. Dieser Baustein ist so schnell, daß eine Rückmeldung zum Prozessor entfallen kann. Über einen schnellen Operationsverstärker wird das vom D/A-Umsetzer gelieferte Stromsignal in ein Spannungssignal umgewandelt.

Abb. 1.3.5-2 zeigt eine weitere Schaltung für den D/A-Umsetzer. Wird das analoge Signal z.B. vom Computer zum Oszilloskop über eine längere Strecke geführt, so ist es unvermeidlich, daß Störungen eingefangen werden, insbesondere aufgrund der höherfrequenten Schaltvorgänge des Computers. Die abgebildete Schaltung ist in der Lage, das hochfrequente Rauschen durch die geringe Bandbreite des verwendeten Operationsverstärkers LM 3900 wieder zu entfernen, ohne daß das Signal darunter merklich leidet. Die Schaltung muß aber mit einer getrennten Stromversorgung betrieben werden.

Der Operationsverstärker ist mit den Widerständen R1 und R2 als 1:1-Verstärker geschaltet, wobei der gestrichelt eingezeichnete Widerstand R3 zur Potentialverschiebung dient, so daß der Operationsverstärker mit nur 5 V betrieben werden kann. Anwendung des D/A-Umsetzers sind die Verwendung als digitaler Funktionsgenerator, in Verbindung mit einem A/D-Umsetzer als Speicherscop, ferner Steuerung von analogen Geräten, Musikerzeugung und Sprachwiedergabe.

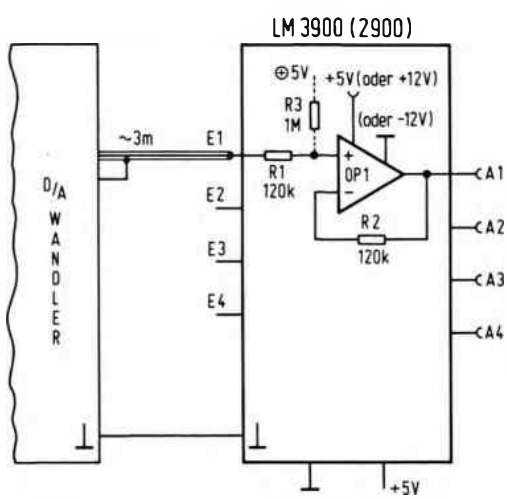
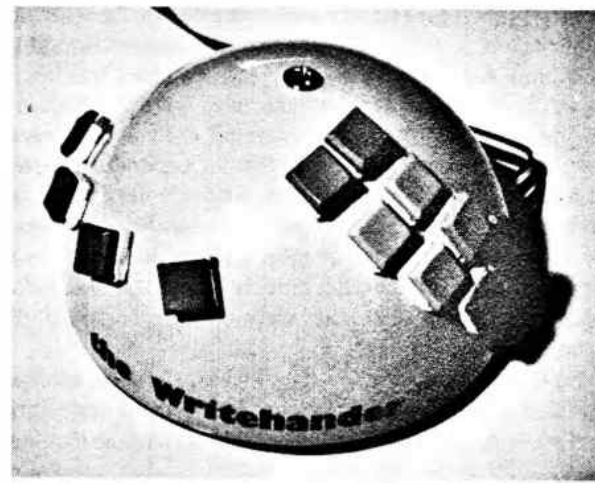


Abb. 1.3.5-2 Regenerierverstärker für den D/A-Umsetzer

1.4 Besondere Peripheriesysteme

An dieser Stelle werden ein paar Peripherieeinheiten besprochen, die eine reizvolle Ergänzung des Computers sein können.



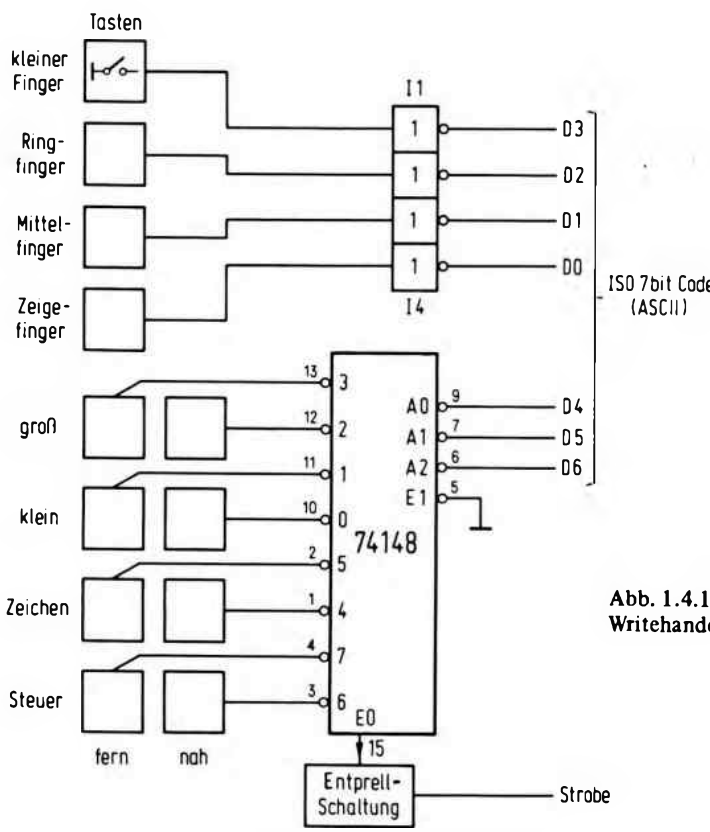


Abb. 1.4.1-1 Schaltbild des Writehanders

1.4.1 Writehander

Foto 1.4.1-1 zeigt die Gestalt des sogenannten Writehanders. Es handelt sich dabei um eine alphanumerische Tastatur. *Abb. 1.4.1-1* zeigt das Schaltbild des Innenlebens. Das Gerät wurde in Amerika entwickelt und kann als Tastaturersatz verwendet werden. Der Vorteil im Gegensatz zur konventionellen Tastatur liegt darin, daß der Writehander mit einer Hand bedient werden kann. Die andere Hand bleibt dann frei und kann zum Beispiel zum Verfolgen eines Listings verwendet werden, das in den Computer eingetastet werden soll. Der Writehander wird wie folgt bedient: Kleine Finger, Ringfinger, Mittelfinger und Zeigefinger werden auf die vier abgelegenen Tasten gelegt. Mit dem Daumen wird eine der acht paarweise angeordneten Tasten betätigt. Die vier Finger müssen einen Code direkt eingeben, und mit dem Daumen wird eine von 16 Codegruppen ausgewählt. Der Takt wird dabei mit dem Daumen erzeugt, so daß der Code zuerst mit den vier anderen Fingern eingestellt wird. Der Code ist eingeteilt in vier Klassen: Großbuchstaben, Kleinbuchstaben, Sonderzeichen und Steuerzeichen. Die Klassen sind dann noch in zwei Gruppen eingeteilt, die mit der Tastenreihe „fern" und „nah" ausgewählt werden.

Der „Tastaturencoder" ist dabei recht einfach aufgebaut, da die eigentliche ASCII-Codierung der Benutzer durch Betätigen der richtigen Tasten selbst vornehmen muß. *Abb. 1.4.1-2* zeigt nochmals die vollständige Codetabelle für die Tastatur.

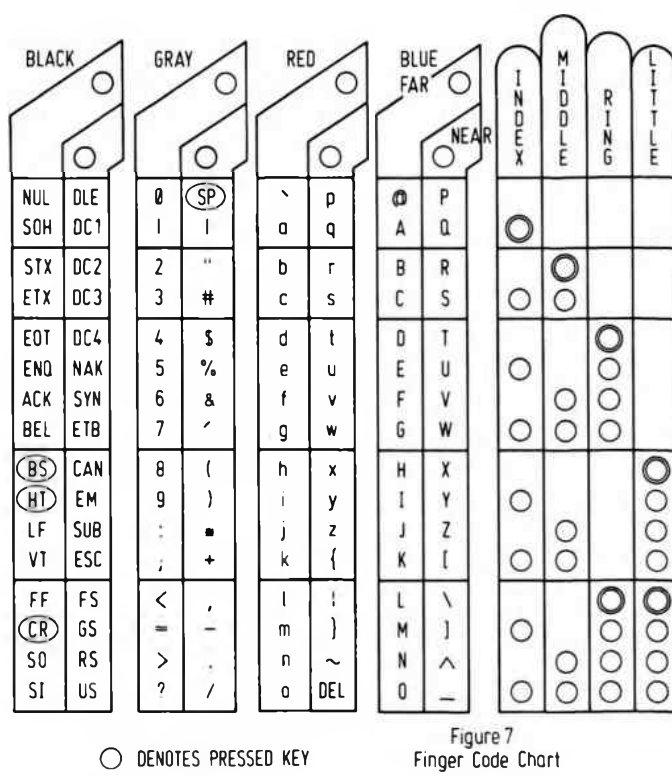


Abb. 1.4.1-2 Codetabelle der Tastatur

Der Leser wird sich nun fragen, wie die Erlernung der Bedienung des Writehanders überhaupt möglich ist? Dazu gibt es ein paar Tricks, die es gestatten, die Tastatur schon nach wenig Übungszeit ohne Codetabelle zu betätigen. Es wird zunächst die erste Gruppe erlernt. Der Daumen tastet dazu die Taste „Großbuchstaben fern". Es wird der Zeigefinger betätigt und danach der Daumen. Es erscheint der Buchstabe A. Bei Betätigen des Mittelfingers anstatt des Zeigefingers erscheint B, bei Ringfinger D und beim kleinen Finger H. Wird der Ringfinger und der kleine Finger gleichzeitig getastet, so erscheint L. Diese fünf Zeichen müssen auswendig gelernt werden: ABDHL. Das gleiche gilt für die Gruppe „Großbuchstaben nah". Dort sind es die Buchstaben QRTX . Ausgehend von diesen beiden Buchstabengruppen kann durch Abzählen ein davor oder dahinter liegende Buchstabe ermittelt werden. Für die Klasse „Kleinbuchstaben" ist die Codierung genau gleich.

Mit Hilfe von Eselsbrücken und etwas Übung kann eine relativ hohe Schreibgeschwindigkeit erreicht werden, die diese Tastatur zu einem interessanten Eingabegerät werden läßt.

1.4.2 Digitalisierer

Ein Digitalisierer dient der Umsetzung einer analogen Position in einen digitalen Wert. Die Position kann zum Beispiel mit einem Griffel auf einer speziellen Platte bestimmt werden. *Foto 1.4.2-1* zeigt einen solchen Digitalisierer.

Diese Geräte waren bis vor kurzem noch sehr teuer. Doch heute wird z.B. von Summagraphics auch ein preiswertes Gerät (unter 2000,- DM) angeboten, daß eine Auflösung von 1/10 mm bei einer gesamten Fläche von 300 mm x 300 mm besitzt.

Abb. 1.4.2-1 zeigt das grobe Blockschaltbild dieses Digitalisierers. Es ist mit einem Mikrocomputer aufgebaut, der den internen Ablauf steuert, ferner aus einer Zählerkette, sowie Treiberstufen. Die Ausgänge der Treiberstufen sind an ein gitterförmiges Netz angeschlossen, das sich unter der Digitalisierungsfläche befindet.

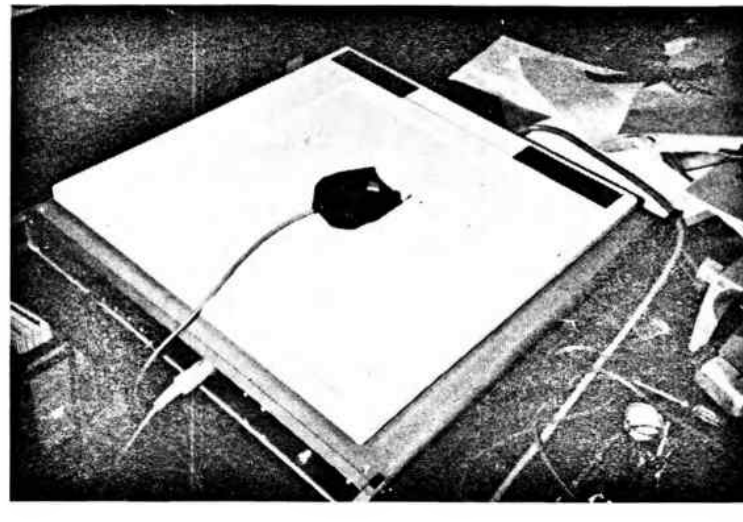


Abb. 1.4.2-1 Blockschaltbild eines Digitalisierers

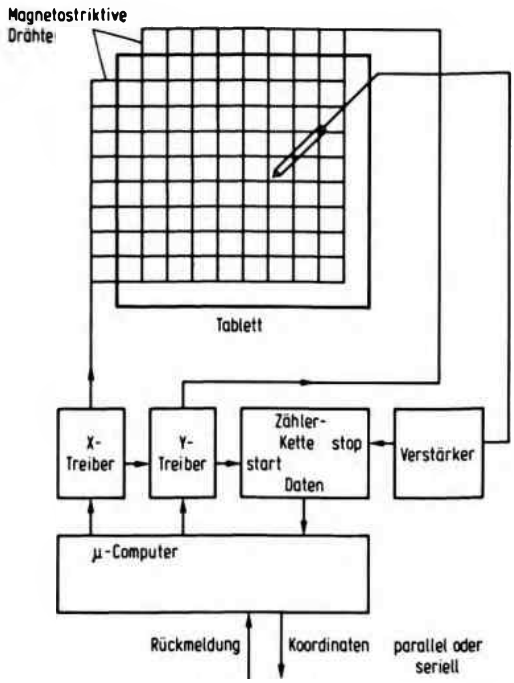


Abb. 1.4.2-1 Blockschaltbild eines Digitalisierers

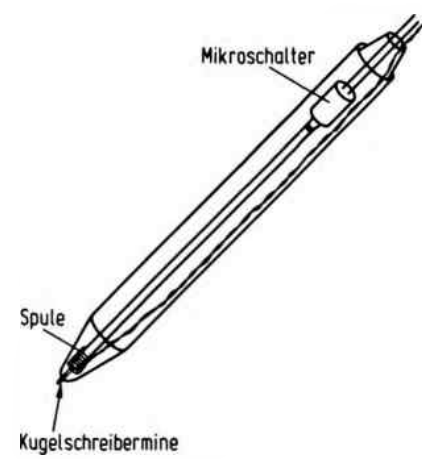


Abb. 1.4.2-2 Griffel als Aufnehmer

Wird z.B. ein Impuls auf die X-Treiber gegeben, so läuft eine Art mechanischer Welle, begleitet von einem elektromagnetischen Feld, durch die magnetostriktiven Drähte der X-Richtung. Ein kleiner Empfänger, bestehend aus einer Spule, die an dem Griffel befestigt ist, nimmt das Signal auf. Nach Auflösen des Impulses wird eine Zählerkette gestartet, sobald die Welle den Griffel erreicht hat, wird die Zählerkette gestoppt. Der Zählerinhalt ist damit ein direktes Maß für die Position des Griffels. Der gleiche Vorgang wird anschließend für die Y-Leitung vorgenommen. Als Ergebnis stehen zwei Koordinaten zur Verfügung, die dem Hauptcomputer übermittelt werden. *Abb. 1.4.2-2* zeigt den prinzipiellen Aufbau des Griffels. Neben der Empfängerspule ist auch noch ein Mikroschalter in das Griffelgehäuse eingebaut, der immer dann betätigt wird, wenn die Digitalisierungsoberfläche berührt wird. *Abb. 1.4.2-3* zeigt eine andere Ausführungsform eines Aufnehmers. Der Aufnehmer ist mit einem Fadenkreuz zur genauen

Positionierung ausgestattet. Um das Fadenkreuz herum ist eine Spule eingelassen, die den Positionsimpuls aufnimmt. Mit einer Taste kann das Erreichen der Position an den Rechner gemeldet werden.

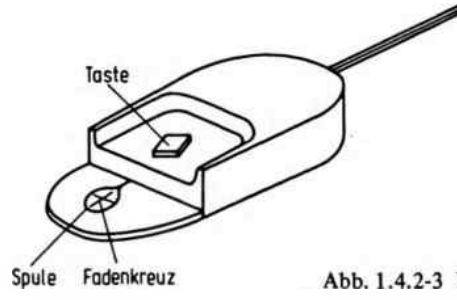


Abb. 1.4.2-3 Fadenkreuzaufnehmer

2 Software

2.1 Monitorprogramme

Nach dem Einschalten eines Mikrorechners startet dieser die Ausführung eines Programms an einer bestimmten Speicherstelle. Um von außen mit diesem Mikrorechner in Verbindung treten zu können, muß ein spezielles Programm (das „Monitorprogramm") an diese Stelle gelegt werden. Das Monitorprogramm bewirkt zunächst eine Initialisierung aller IO-Geräte und meldet sich dann – meist über eine Konsole – beim Benutzer. Dem Rechner können dann mit Hilfe des Monitorprogramms Befehle gegeben werden, z.B. um Programme einzulesen oder den Inhalt von Speicherzellen zu betrachten und zu modifizieren. Es kann auch ein „Bootstrap" gestartet werden, der ein größeres Betriebsprogramm einer Floppy-Einrichtung lädt.

Hier soll ein kleiner Monitor besprochen werden, der es erlaubt, Programme einzugeben und zu modifizieren. Er kann damit für die Programmentwicklung nützlich sein.

2.1.1 Befehle des Monitors

Der Monitor versteht drei Grundbefehle, die nachstehend zusammengestellt sind:

G : GO
Sprung nach einer Speicherzelle und Ausführung des dort stehenden Maschinenprogramms. Beispiel: G1ØØcr Sprung zur Adresse 1ØØH (cr bedeutet „carriage return").

Q : QUERRY
Damit kann der Zustand von IO-Kanälen abgefragt und verändert werden. QI bewirkt die Eingabe von einem Kanal; z.B. QI1Øcr gibt den am Kanal 1ØH stehenden Wert ein. Q0 gibt einen Wert an einen IO-Kanal; z.B. Q02Ø33cr gibt den Wert 33H an den Kanal 2ØH.

E : ENTER
E1ØØcr gibt den Speicher 1ØØ zum Modifizieren frei. Danach gibt es mehrere weitere Befehle, die den Ablauf steuern.

rub Um eine Zelle rückwärts gehen (rub hat den Code 7FH).
cr Um eine Zelle vorwärts gehen.
blank Um einen Befehl weiter. Dabei wird der Inhalt der aktuellen Speicherzelle als Z80-Befehl interpretiert. Ist der Befehl z.B. 3 Byte lang, so wird der Adreßzähler um 3 Byte erhöht.
. Ende des ENTER-Modus.
" Eingabe eines Textes mit Code nach DIN 66003 (ASCII). Der Text wird mit " abgeschlossen. Bei Eingabe von rub rückt der Cursor um eins nach links und das zuletzt eingegebene Zeichen wird wieder gelöscht.

Abb. 2.1.1-1 zeigt ein Beispiel für die Bedienung des Monitors.

```
RDK MONITOR V1.0
>E100

0100 C3 29 02    :C:
0103 C3 03 F0    :C:
0106 C3 0F F0    :C:
0109 C3 12 F0    :C:
010C C3 1E F0    :C:
010F F5          :U:
0110 1F          : :
0111 1F          : :
0112 1F          : :
0113 1F          : :
0114 CD 18 01    :M:
0117 F1          :Q:
0118 E6 0F       :F:.
RDK MONITOR V1.0
>
```

Abb. 2.1.1-1 Beispiel für die Bedienung des Monitors

```
RDK MONITOR V1.0
>QI10
 0D
RDK MONITOR V1.0
>QO10 23
```

```
RDK MONITOR V1.0
>GF01E
```

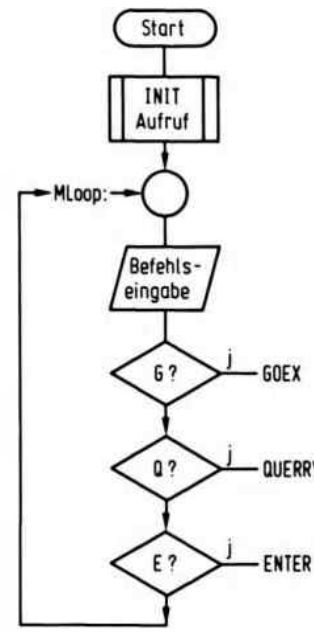


Abb. 2.1.2-1 Flußdiagramm des Hauptprogramms

2.1.2 Funktionsweise des Monitorprogramms

Abb. 2.1.2-1 zeigt das Flußdiagramm des Hauptprogramms. Zuerst werden IO-Kanäle mit dem Aufruf CALL INIT initialisiert. Es werden z.B. Baud-Raten mit jeweils zugehörigem UART eingestellt. Danach meldet sich das System und wartet auf eine Befehlseingabe. Wird ein Zeichen eingegeben, wie G, Q oder E, so springt das Programm an die Stellen GOEX, QUERRY oder ENTER. Andernfalls wird wieder auf eine Eingabe gewartet.

Abb. 2.1.2-2 zeigt die Programme GOEX und QUERRY. Bei GOEX wird eine Sprungadresse von der Konsole geholt. Dann wird an die entsprechende Stelle gesprungen. QUERRY erwartet zunächst noch ein weiteres Zeichen, um zu entscheiden, ob es sich um die Eingabe eines Wertes oder die Ausgabe auf einen Kanal handelt. Beim Zeichen I soll ein Wert von dem IO-Kanal geholt werden, und die Port-Adresse muß über die Konsole eingegeben werden. Danach wird der IO-Kanal angesprochen und der Wert der Konsole angezeigt. Bei dem Zeichen 0 wird ebenfalls eine Port-Adresse von der Konsole geholt, und dazu noch ein Ausgabewert. Dieser Wert wird dann auf den angegebenen Kanal ausgegeben.

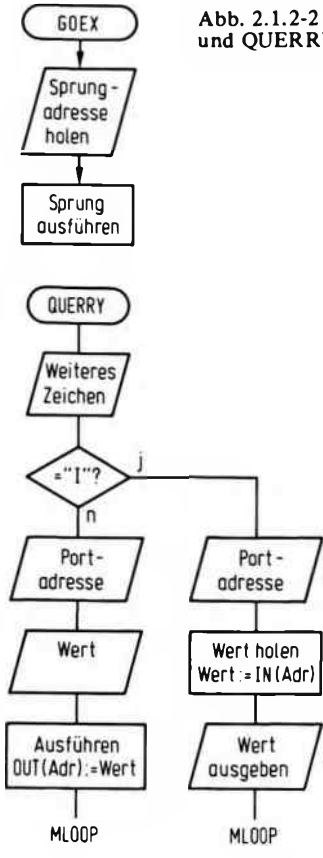


Abb. 2.1.2-2 Teilprogramme GOEX und QUERRY

Abb. 2.1.2-3 Teilprogramm ENTER

Abb. 2.1.2-3 zeigt das Teilprogramm ENTER. Es wird in jedem Fall eine Adresse von der Konsole geholt. Nach einem Zeilenvorschub wird die angegebene Adresse noch einmal ausgegeben. Anschließend wird die Befehlslänge das auf dieser Adresse stehende Byte berechnet. Je nach Befehlslänge werden die erforderlichen weiteren Speicherzellen abgefragt und deren Inhalte ausgegeben. Nach Abschluß dieser Prozedur wird auf die Eingabe eines weiteren Befehls gewartet. *Abb. 2.1.2-4* zeigt den Entscheidungsablauf. Bei Eingabe des Zeichens Rubout wird die Adresse um eins decrementiert, bei cr wird die Adresse um eins incrementiert. Wurde ein Leerzeichen blank eingegeben, so wird die Adresse um die Anzahl der Zeichen der Befehlslänge incrementiert.

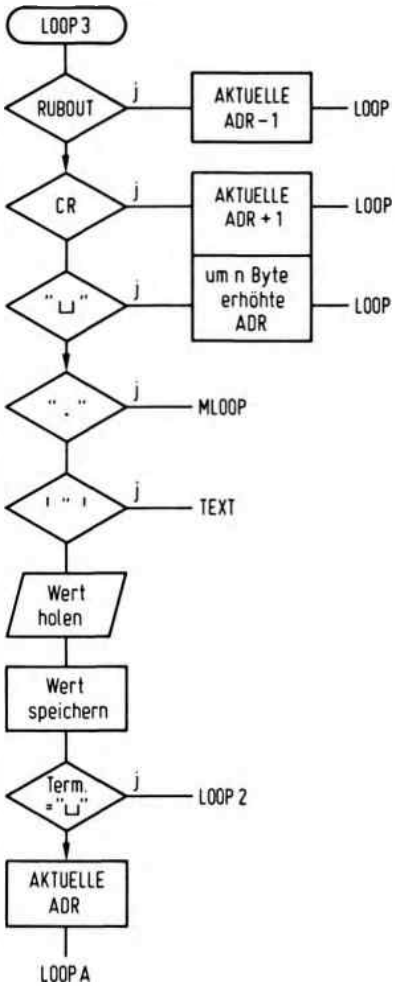


Abb. 2.1.2-4 Entscheidungsablauf in LOOP3

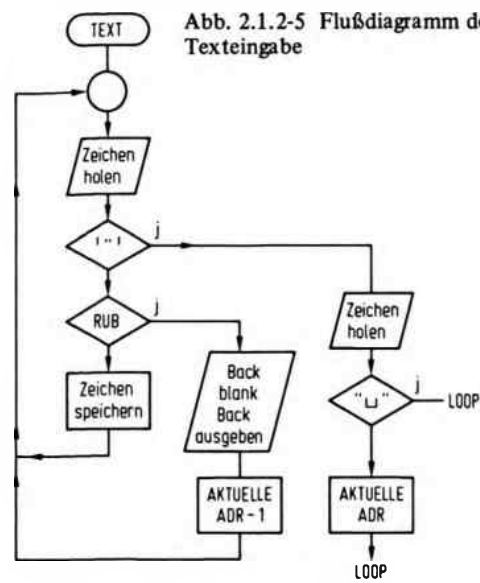


Abb. 2.1.2-5 Flußdiagramm der Texteingabe

So entsteht im Konsolbild eine Art Pseudodisassamblierung. Bei "." wird wieder zum Hauptprogramm gesprungen. Wird " eingegeben, so erfolgt ein Sprung zur Texteingabe.

Andernfalls kann ein Wert eingegeben werden, der dann auf der gerade adressierten Speicherzelle abgelegt wird. Nach Eingabe von blank kann ein weiterer Wert angegeben werden, der auf der nächsten Zelle abgelegt wird. Dies geschieht solange, bis cr eingegeben wird, dann wird die nächste Adresse angezeigt. *Abb. 2.1.2-5* zeigt das Flußdiagramm der Texteingabe. Zunächst wird ein Zeichen vom Benutzer eingegeben. Ist es das Zeichen ", so wird die Texteingabe abgeschlossen. Andernfalls wird überprüft, ob es sich um das Zeichen rubout handelt. Ist das der Fall, so wird der Adreßzähler um eins decrementiert und die Zeichenfolge back blank back ausgegeben, die auf der Konsole das zuvor eingegebene Zeichen löscht. In jedem anderen Fall wird das Zeichen als

DIN 66003-Zeichen (ASCII) interpretiert und im Speicher abgelegt. Der Adreßzähler wird um eins incrementiert. Nach Abschluß der Texteingabe wird ein weiteres Zeichen geholt. Dabei wird unterschieden, ob das Zeichen blank eingegeben wurde oder nicht. Bei blank bleibt der Cursor in der gleichen Zeile, es können dann noch andere Bytes eingegeben werden.

2.1.3 Realisierung des Programms

Abb. 2.1.3-1 zeigt das vollständige „Listing" des beschriebenen Monitorprogramms. Das Listing wurde mit dem TDL-Diskassembler erstellt. Es beginnt mit den Pseudobefehlen (d.h. Steueranweisungen an den Assembler). .PHEX und .PREL. Der Befehl .PHEX bewirkt, daß der Objekt-Code im ASCII HEX (Sedezimal nach DIN 66003 mit den Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F mit jeweils 7 Bit) Format abgelegt wird und somit ausdruckbar ist. Der Befehl .PREL bewirkt, daß der Objekt-Code relokalisierbar, also verschiebbar ist.

Der nächste Steuerbefehl ist .INSERT A:IOPACK. Er bewirkt, daß ein auf der Floppy Disk vorhandenes Programm in das Gesamtprogramm mit einbezogen wird. Damit ist es möglich, immer wieder benötigte Programmteile in ein Hauptprogramm einzufügen, ohne sie jedesmal neu schreiben zu müssen. Der Befehl A:IOPACK bedeutet, daß das Programm mit dem Namen IOPACK.ASM (.ASM ist voreingestellt) in das Gesamtprogramm eingefügt wird. Der Vorteil gegenüber dem sogenannten Binden (linking) von Programmen liegt darin, daß ein LINKING loader entfallen kann, der das Programm nach der Übersetzung mit dem Teilprogramm zusammenfügt.

Das Teilprogramm IOPACK beinhaltet Ein- und Ausgabedefinitionen, sowie Routinen zur Eingabe von mehreren Werten, die dann sedezimal interpretiert werden. Sie können so vom 7-Bit-Code (ASCII) in das interne Format übersetzt werden (EXPR). Im Sedezimalformat sind noch Routinen zur Ausgabe des Akkumulatorinhaltes (PRAC) und des HL-Registers (PRHL). Eine Besonderheit stellt eine Makrodefinition von PRINT mit dem Pseudobefehl .DEFINE dar. PRINT kann im Programm verwendet werden, um 7-Bit-codierte (ASCII)-Texte auf der Konsole auszugeben, z.B. PRINT „TEXT", d.h. es soll Text auf der Konsole ausgedruckt werden. PRINT enthält zwei Parameter. Der erste Parameter mit dem Namen A dient zur Eingabe des auszudruckenden Textes, %B ist ein Pseudoparameter und wird beim Aufruf nicht angegeben. %B bewirkt, daß bei jedem erneuten Aufruf des Makros ein anderer Wert für %B erzeugt wird. Dies ist erforderlich, weil %B innerhalb des Makros als Marke vorkommt. Wenn der Makro mehrmals aufgerufen werden würde, so würden bei Verwendung einer normalen Marke Doppeldefinitionen entstehen.

Eine interessante Routine ist das Unterprogramm LENGTH. Es bestimmt die Länge eines Z80-Befehls. Beim Aufruf muß in Hl die Adresse des Befehls stehen, dessen Länge bestimmt werden soll. In B wird nach Aufruf die errechnete Länge übergeben. Das Unterprogramm behandelt auch alle bekannten Pseudobefehle des Z80-Befehlssatzes richtig.

Das Hauptprogramm beginnt bei Adresse 129H und läuft entsprechend der vorhergehenden Flußdiagramme ab. Um den Monitor an ein bestimmtes System anzupassen, ist erstens die Routine INIT zu schreiben. Auf Adresse 164H sind 2ØH Byte für diesen Zweck freigelassen. Zweitens müssen noch die Vektoren für die Routinen CI, CO und CSTS geändert werden. Bei CI darf kein Register außer dem Akku verändert werden.

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 1
.MAIN. -


                        .PHEX
                        .PREL
                        ;
                        ;
                        ;
                        ;*************************************************
                        ;* R D K  M O N I T O R  V790603 V1.0           *
                        ;*************************************************
                        ;
                        ;
                        ;
                        .INSERT A:IOPACK        ;STANDARD ROUTINEN
                       @;*********************************
                       @;* INCLUDE FILE
                       @;* MIT .INSERT IOPACK
                       @;* RDK 790519
                       @;* BEINHALTET STANDART IO + HEX-
                       @;* UMRECHNUNG
                       @;*********************************
                       @
0000'   C3 0129'       @JMP START          ;START DES HAUPTPROGRAMMES
0003'   C3 F003        @CI:JMP 0F003H
0006'   C3 F009        @CO:JMP 0F009H
0009'   C3 F012        @CSTS: JMP 0F012H
000C'   C3 F01E        @EXIT: JMP 0F01EH
                       @;
                       @; ROUTINEN EXPR, PRAC,PRHL
                       @; SOWIE CRLF UND PRINT
                       @;
000F'                  @PRAC:              ;GIBT A IN ZWEI DIGITS AUS
000F'   F5             @PUSH PSW
0010'   1F             @RAR
0011'   1F             @RAR
0012'   1F             @RAR
0013'   1F             @RAR
0014'   CD 0018'       @CALL OUTH
0017'   F1             @POP PSW
0018'                  @OUTH:
0018'   E60F           @ANI 0FH
001A'   C630           @ADI '0'
001C'   FE3A           @CPI '9'+1
001E'   DA 0023'       @JC OUTCH
0021'   C607           @ADI 'A'-'9'-1
0023'                  @OUTCH:
0023'   4F             @MOV C,A
0024'   C3 0006'       @JMP CO
                       @;
                       @;
0027'                  @PRHL:              ;GIBT HL IN 2 BYTES (4DIGITS )AUS
0027'   7C             @MOV A,H
0028'   CD 000F'       @CALL PRAC
002B'   7D             @MOV A,L
002C'   18E1           @JMPR PRAC
                       @;
                       @;
```

Abb. 2.1.3-1 Listing des Monitorprogramms

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 2
.MAIN. -


                        @; EXPR HOLT ZEICHEN VON DER CONSOLE
                        @; UND SPEICHERT DAS ERGEBNIS IN HL
                        @; DIE EINGABE IST FORMATFREI
                        @; DAS TERMINALZEICHEN WIRD IN A UEBERGEBEN
                        @;
   002E'                @EXPR:
   002E'  21 0000       @LXI H,0            ;ANFANGSWERT
   0031'                @EX0:
   0031'  CD 0003'      @CALL CI
   0034'  CD 0023'      @CALL OUTCH
   0037'                @EX1:
   0037'  CD 0045'      @CALL NIBBLE
   003A'  DA 0055'      @JC EX2
   003D'  29            @DAD H                              zu Abb. 2.1.3-1
   003E'  29            @DAD H
   003F'  29            @DAD H
   0040'  29            @DAD H
   0041'  B5            @ORA L
   0042'  6F            @MOV L,A
   0043'  18EC          @JMPR EX0
   0045'                @NIBBLE:
   0045'  D630          @SUI '0'
   0047'  D8            @RC
   0048'  FE17          @CPI 'G'-'0'
   004A'  3F            @CMC
   004B'  D8            @RC
   004C'  FE0A          @CPI 10
   004E'  3F            @CMC
   004F'  D0            @RNC
   0050'  D607          @SUI 'A'-'9'-1
   0052'  FE0A          @CPI 10
   0054'  C9            @RET
   0055'                @EX2:
   0055'  79            @MOV A,C            ;TERMINATOR
   0056'  FE0D          @CPI 0DH
   0058'  CA 005C'      @JZ ECHO
   005B'  C9            @RET
                        @;
   005C'                @ECHO:
   005C'  F5            @PUSH PSW
   005D'  0E0A          @MVI C,0AH
   005F'  CD 0006'      @CALL CO
   0062'  F1            @POP PSW
   0063'  4F            @MOV C,A
   0064'  C9            @RET
                        @;
                        @;
   0065'                @CRLF:              ;GIBT CRLF AUF DER CONSOLE AUS
   0065'  0E0D          @MVI C,0DH
   0067'  CD 0006'      @CALL CO
   006A'  0E0A          @MVI C,0AH
   006C'  1898          @JMPR CO
                        @;
                        @;
   006E'                @LPRINT:            ;DRUCKT EINEN TEXT BIS 0 AUS
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 3
.MAIN. -


                        @                       ;ADRESSE IN HL
006E'   7E              @MOV A,M
006F'   23              @INX H
0070'   B7              @ORA A
0071'   C8              @RZ
0072'   CD 0023'        @CALL OUTCH
0075'   18F7            @JMPR LPRINT
                        @.DEFINE PRINTCA,%B]=
                        @[
                        @LXI H,.+8                              zu Abb. 2.1.3-1
                        @CALL LPRINT
                        @JMPR %B
                        @.ASCIZ A
                        @%B:
                        @]
                        @;
                        @; ******** ENDE IOPACK 790519 *********
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ; DIE NACHFOLGENDE ROUTINE BESTIMMT
                         ; DIE LAENGE EINES Z80 BEFEHLS
                         ; HL ZEIGT AUS DEN BEFEHL UND IN B
                         ; WIRD DIE ANZAHL DER BYTES UEBERGEBEN
                         ;
                         ;
0077'                    LENGTH:
0077'   0600            MVI B,0             ;BYTEZAEHLER
0079'   E5              PUSH H
007A'   D1              POP D               ;HL NACH DE RETTEN
007B'   7E              MOV A,M             ;BYTE HOLEN
007C'   E6DF            ANI 0DFH            ;DD FD BEFEHL MASKIEREN
007E'   FEDD            CPI 0DDH            ;UMSCHALTBEFEHL?
0080'   CA 00ED'        JZ TAB5             ;DD FD TABELLE
0083'   7E              MOV A,M             ;WERT WIEDER HOLEN
0084'   FECB            CPI 0CBH            ;CB TABELLE?
0086'   CA 00E7'        JZ TAB3             ;DANN DORTHIN
0089'   FEED            CPI 0EDH            ;ED TABELLE
008B'   CA 00DB'        JZ TAB2             ;OK DORTHIN
008E'   7E              MOV A,M
008F'   FEC3            CPI 0C3H            ;JMP BEFEHL
0091'   CA 0124'        JZ B3               ;3 BYTES
0094'   FECD            CPI 0CDH            ;CALL BEFEHL
0096'   CA 0124'        JZ B3               ;3 BYTES
0099'   E6EF            ANI 0EFH            ;NEUE BEFGRUPPE
009B'   FE22            CPI 22H             ;LD (NN),XX
009D'   CA 0124'        JZ B3               ;3 BYTES
00A0'   FE2A            CPI 2AH             ;LD XX,(NN)
00A2'   CA 0124'        JZ B3               ;3 BYTES
00A5'   E6CF            ANI 0CFH            ;NEUE MASKE
00A7'   FE01            CPI 1               ;LD XX,NN
00A9'   CA 0124'        JZ B3               ;3 BYTES
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. --

00AC'   E6C7        ANI 0C7H        ;MASKE
00AE'   FEC2        CPI 0C2H        ;JP COND
00B0'   CA 0124'    JZ B3           ;3 BYTES
00B3'   FEC4        CPI 0C4H        ;CALL COND
00B5'   CA 0124'    JZ B3           ;3 BYTES
00B8'   7E          MOV A,M         ;2 BYTE BEFEHLE
00B9'   E6F7        ANI 0F7H        ;NEUE MASKE
00BB'   FE10        CPI 10H         ;DJNZ,JR
00BD'   CA 0125'    JZ B2           ;2 BYTES
00C0'   FED3        CPI 0D3H        ;IN,OUT
00C2'   CA 0125'    JZ B2           ;2 BYTES
00C5'   E6E7        ANI 0E7H        ;MASKE
00C7'   FE20        CPI 20H         ;JR COND
00C9'   CA 0125'    JZ B2           ;2 BYTES
00CC'   E6C7        ANI 0C7H        ;NEUE MASKE
00CE'   FE06        CPI 6           ;LD,N
00D0'   CA 0125'    JZ B2           ;2 BYTES
00D3'   FEC6        CPI 0C6H        ;OP,N
00D5'   CA 0125'    JZ B2           ;2 BYTES
00D8'   C3 0126'    JMP B1          ;REST SIND 1 BYTE BEFS
                    ;
                    ;
00DB'               TAB2:
00DB'   23          INX H           ;ED TABELLE, NAECHSTES
00DC'   7E          MOV A,M         ;BYTE UNTERSUCHEN
00DD'   E6C7        ANI 0C7H        ;MASKE
00DF'   FE43        CPI 43H         ;LD (NN),XX LD XX,(NN)
00E1'   CA 0123'    JZ B4           ;4 BYTE BEFEHLE
00E4'   C3 0125'    JMP B2          ;REST 2 BYTE BEFEHLE
                    ;
                    ;
00E7'               TAB3:
00E7'   C3 0125'    JMP B2          ;ALLES 2 BYTE BEFEHLE
                    ;
                    ;
00EA'               TAB4:
00EA'   C3 0123'    JMP B4          ;ALLES 4 BYTE BEFEHLE
                    ;
                    ;
00ED'               TAB5:
00ED'   23          INX H           ;DD FD TAB  NAECHSTER WERT
00EE'   7E          MOV A,M         ;HOLEN
00EF'   FECB        CPI 0CBH        ;DD FD - CB TABELLE
00F1'   CA 00EA'    JZ TAB4         ;JA DORTHIN
00F4'   FE21        CPI 21H         ;LD II,NN
00F6'   CA 0123'    JZ B4           ;4 BYTES
00F9'   E6FE        ANI 0FEH        ;MASKE
00FB'   FE34        CPI 34H         ;INC DEC (II)
00FD'   CA 0124'    JZ B3           ;3 BYTES
0100'   E6F8        ANI 0FBH
0102'   FE70        CPI 70H         ;LD (II),R
0104'   CA 0124'    JZ B3           ;3 BYTES
0107'   7E          MOV A,M         ;NOCHMAL HOLEN
0108'   E6CF        ANI 0CFH
010A'   FE06        CPI 6           ;LD (II),N
```

zu Abb. 2.1.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 5
.MAIN. -


010C'   CA 0123'     JZ B4              ;4 BYTES
010F'   E6C7         ANI 0C7H
0111'   FE02         CPI 2              ;LD (NN),II
0113'   CA 0123'     JZ B4              ;4 BYTES
0116'   7E           MOV A,M            ;WERT HOLEN
0117'   D640         SUI 40H            ;FUER VERGLEICH
0119'   E687         ANI 87H
011B'   FE06         CPI 6              ;LD R,(II)
011D'   CA 0124'     JZ B3              ;3 BYTES
0120'   C3 0125'     JMP B2             ;REST 2 BYTE INCL PSEUDOBEF
                     ;
                     ;
0123'   04           B4:INR B
0124'   04           B3:INR B
0125'   04           B2:INR B
0126'   04           B1:INR B
0127'   EB           XCHG               ;HL WIEDER ZURUECK
0128'   C9           RET                ;FERTIG
                     ;
                     ;
                     ;******* S T A R T   D E S   H A U P T P R O G R A M M S
                     ***
                     ;
0129'                START:
0129'   CD 0164'     CALL INIT          ;IO INITIALISIEREN  BENUTZERROUTINE
012C'                MLOOP:
                     ;
012C'   CD 0065'     CALL CRLF
                     PRINT 'RDK MONITOR V1.0'[0DH] [0AH] '>'[
012F'   21 0137'    +LXI H,.+8
0132'   CD 006E'    +CALL LPRINT
0135'   1814        +JMPR ..0001
0137'   52444B204D4F+.ASCIZ 'RDK MON\
013D'   4E49544F5220+\ITOR V\
0143'   56312E300D0A+\1.0'[0DH] [0AH] '>'\
0149'   3E00        +\
                    +]
014B'   CD 0003'     CALL CI
014E'   4F           MOV C,A
014F'   CD 0006'     CALL CO            ;BEFEHLS EINGABE
0152'   FE45         CPI 'E'
0154'   CA 0188'     JZ ENTER           ;EINGABE VON BYTES ..
0157'   FE47         CPI 'G'            ;PROGRAMM AUSFUEHREN
0159'   CA 0185'     JZ GOEX
015C'   FE51         CPI 'Q'            ;IO DEBUG
015E'   CA 0189'     JZ QUERRY
0161'   C3 012C'     JMP MLOOP
                     ;
0164'   C9           INIT: RET          ;KURZGESCHLOSSEN
0165'                .BLKB 20H
                     ;
                     ;
                     ;
                     ;
                     ; * UNTERPROGRAMME *
```

zu Abb. 2.1.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 6
.MAIN. -


                        ;
                        ;
0185'                   GOEX:
0185'   CD 002E'       CALL EXPR       ;SPRUNG AUF USER PROGRAMM
0188'   E9              PCHL
                        ;
0189'                   QUERRY:
0189'   CD 0003'       CALL CI
018C'   CD 0023'       CALL OUTCH      ;QI ODER QO
018F'   FE49            CPI 'I'
0191'   2814            JRZ INPQ
0193'   CD 002E'       CALL EXPR       ;QO AUSGABE PORT
0196'   4D              MOV C,L         ;ADRESSE
0197'   FE20            CPI ' '         ;TERMINATOR
0199'   C2 012C'       JNZ MLOOP
019C'   C5              PUSH B          ;PORT RETTEN
019D'   CD 002E'       CALL EXPR       ;WERT
01A0'   C1              POP B           ;PORT ADR ZURUECK
01A1'   7D              MOV A,L
01A2'   ED79            OUTP A          ;OUT (C),A
01A4'   C3 012C'       JMP MLOOP
                        ;
01A7'                   INPQ:
01A7'   CD 002E'       CALL EXPR       ;PORT NR
01AA'   0E20            MVI C,' '
01AC'   CD 0006'       CALL CO
01AF'   4D              MOV C,L
01B0'   ED78            INP A
01B2'   CD 000F'       CALL PRAC       ;AUSGEBEN
01B5'   C3 012C'       JMP MLOOP
                        ;
01B8'                   ENTER:          ;EINGABE VON BYTES
                        ;
01B8'   CD 002E'       CALL EXPR       ;ADRESSE HOLEN NACH HL
                        ;
01BB'                   LOOP:
01BB'   CD 0065'       CALL CRLF
01BE'                   LOOPA:
                        ;
01BE'   CD 0027'       CALL PRHL       ;AUSGEBEN DER ADRESSE
01C1'   CD 0077'       CALL LENGTH     ;LAENGE DES BEFEHLS (HL)
                                        ;IN B STEHT DIE ANZAHL
01C4'   C5              PUSH B          ;RETTEN LAENGE
01C5'   E5              PUSH H          ;POINTER RETTEN
01C6'                   LOOP1:
01C6'   0E20            MVI C,' '
01C8'   CD 0006'       CALL CO         ;BLANK
01CB'   7E              MOV A,M
01CC'   CD 000F'       CALL PRAC       ;(HL) AUSGEBEN
01CF'   23              INX H           ;NAECHSTE ZELLE
01D0'   10F4            DJNZ LOOP1
                        ;
01D2'   D1              POP D           ;POINTER ALT IN DE
01D3'   C1              POP B           ;LAENGE
01D4'   3E04            MVI A,4
```

zu Abb. 2.1.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21
.MAIN. -


01D6'   90          SUB B
01D7'   47          MOV B,A
01D8'   07          RLC
01D9'   E60F        ANI 0FH         ;*3
01DB'   80          ADD B
01DC'   47          MOV B,A
01DD'               LPP:
01DD'   0E20        MVI C,' '
01DF'   CD 0006'    CALL CO
01E2'   10F9        DJNZ LPP
                    ;
                    ;
01E4'   0E3A        MVI C,':'
01E6'   CD 0006'    CALL CO
01E9'   1A          LDAX D          ;ASCII INTERPRETATION
01EA'   E67F        ANI 07FH
01EC'   FE20        CPI 20H
01EE'   DA 01FA'    JC CONT
01F1'   FE7F        CPI 7FH
01F3'   CA 01FA'    JZ CONT
01F6'   4F          MOV C,A
01F7'   C3 01FC'    JMP CONT1
01FA'               CONT:
01FA'   0E20        MVI C,' '
01FC'               CONT1:
01FC'   CD 0006'    CALL CO
01FF'   0E3A        MVI C,':'
0201'   CD 0006'    CALL CO
0204'               LOOP2:
0204'   CD 0003'    CALL CI
0207'   4F          MOV C,A
0208'   CD 0006'    CALL CO         ;BENUTZEREINGABE
020B'               LOOP3:
020B'   FE7F        CPI 07FH
020D'   CA 0238'    JZ BACKGO       ;RUECKWAERTS
0210'   FE0D        CPI 0DH         ;NAECHSTER BEFEHL
0212'   CA 023D'    JZ LPPA
0215'   FE20        CPI ' '         ;INCR UM N BYTES
0217'   CA 01BB'    JZ LOOP
021A'   FE2E        CPI '.'         ;ENDE DER EINGABE
021C'   CA 012C'    JZ MLOOP
021F'   FE22        CPI '"'         ;TEXTEINGABE
0221'   CA 0242'    JZ TEXT
0224'               LOOP4:
0224'   21 0000     LXI H,0
0227'   CD 0037'    CALL EX1
022A'   47          MOV B,A
022B'   7D          MOV A,L
022C'   12          STAX D
022D'   13          INX D
022E'   78          MOV A,B
022F'   FE20        CPI ' '
0231'   CA 0204'    JZ LOOP2
0234'   EB          XCHG
0235'   C3 01BE'    JMP LOOPA
```

zu Abb. 2.1.3-1

```
                        ;
                        ;
0238'                   BACKGO:
0238'   1B              DCX D                          Abb. 2.1.3-1 h
0239'   EB              XCHG
023A'   C3 01BB'        JMP LOOP'
                        ;
                        ;
023D'                   LPPA:
023D'   EB              XCHG            ;HL<->DE
023E'   23              INX H
023F'   C3 01BB'        JMP LOOP
                        ;
0242'                   TEXT:
0242'   CD 0003'        CALL CI
0245'   CD 0023'        CALL OUTCH
0248'   FE22            CPI '"'
024A'   2808            JRZ CONTE
024C'   FE7F            CPI 07FH        ;BACKSPACE
024E'   2813            JRZ BACK
0250'   12              STAX D          ;ABSPEICHERN DES ZEICHENS
0251'   13              INX D           ;ZEICHENZAEHLER
0252'   18EE            JMPR TEXT
                        ;
0254'                   CONTE:
0254'   CD 0003'        CALL CI
0257'   CD 0023'        CALL OUTCH
025A'   FE20            CPI ' '
025C'   CA 0204'        JZ LOOP2        ;WEITER IN DER EINGABE
025F'   EB              XCHG            ;SONST CR O AE
0260'   C3 01BB'        JMP LOOP        ;WEITER MIT CR LF
                        ;
                        ;
0263'                   BACK:
0263'   0E08            MVI C,8
0265'   CD 0006'        CALL CO
0268'   0E20            MVI C,' '
026A'   CD 0006'        CALL CO
026D'   0E08            MVI C,8
026F'   CD 0006'        CALL CO
0272'   1B              DCX D           ;POINTER EINS ZURUECK
0273'   18CD            JMPR TEXT
                        ;
                        .END
```


+++++ SYMBOL TABLE +++++ zu Abb. 2.1.3-1

```
B1      0126'       B2      0125'       B3      0124'       B4      0123'
BACK    0263'       BACKGO  0238'       CI      0003'       CO      0006'
CONT    01FA'       CONT1   01FC'       CONTE   0254'       CRLF    0065'
CSTS    0009'       ECHO    005C'       ENTER   0188'       EX0     0031'
EX1     0037'       EX2     0055'       EXIT    000C'       EXPR    002E'
GOEX    0185'       INIT    0164'       INPQ    01A7'       LENGTH  0077'
LOOP    01BB'       LOOP1   01C6'       LOOP2   0204'       LOOP3   0208'
LOOP4   0224'       LOOPA   01BE'       LPP     01DD'       LPPA    023D'
LPRINT  006E'       MLOOP   012C'       NIBBLE  0045'       OUTCH   0023'
OUTH    0018'       PRAC    000F'       PRHL    0027'       QUERRY  0189'
START   0129'       TAB2    00DB'       TAB3    00E7'       TAB4    00EA'
TAB5    00ED'       TEXT    0242'       .BLNK.  0000:03 X   .DATA.  0000*   X
.PROG.  0275'   X
```

```
!.MAIN.7F
\03.PROG.010000.DATA.020000.BLNK.0300006A
:18010000C32902C303F0C309F0C312F0C31EF0F51F1F1F1FCD1801F1A9
:18011800E60FC630FE3ADA2301C6074FC306017CCD0F017D18E12100D8
:18013000000CD0301CD2301CD4501DA55012929292956F18ECD630D808
:18014800FE173FD8FE0A3FD0D607FE0AC979FE0DCA5C01C9F50E0ACD60
:18016000601F14FC90E0DCD06010E0A18987E2387C8CD230118F7069A
:18017800000E5D17EE6DFFEDDCAED017EFECBCAE701FEEDCADB017EFEDD
:18019000C3CA2402FECDCA2402E6EFFE22CA2402FE2ACA2402E6CFFE39
:1801A80001CA2402E6C7FEC2CA2402FEC4CA24027EE6F7FE10CA2502E5
:1801C000FED3CA2502E6E7FE20CA2502E6C7FE06CA2502FEC6CA250232
:1801D800C32602237EE6C7FE43CA2302C32502C32502C32302237EFE4B
:1801F000CBCAEA01FE21CA2302E6FEFE34CA2402E6F8FE70CA24027EA9
:18020800E6CFFE06CA2302E6C7FE02CA23027ED640E687FE06CA2402A5
:18022000C32502040404EBC9CD6402CD650121370 2CD6E011814529E
:18023800444B204D4F4E49544F52205631 2E300D0A3E00CD03014FCD90
:1502500000601FE45CAB802FE47CA8502FE51CA8902C32C02C9D7
:18028500CD2E01E9CD0301CD2301FE492814CD2E014DFE20C22C02C51B
:18029D00CD2E01C17DED79C32C02CD2E010E20CD06014DED78CD0F012B
:1802B500C32C02CD2E01CD6501CD2701CD7701C5E50E20CD06017ECDE0
:1802CD000F012310F4D1C13E04904707E60F80470E20CD06011 0F90E5B
:1802E5003ACD06011AE67FFE20DAFA02FE7FCAFA024FC3FC020E20CD32
:1802FD000601 0E3ACD0601CD03014FCD0601FE7FCA3803FE0DCA3D033B
:18031500FE20CABB02FE2ECA2C02FE22CA420321 0000CD3701477D12DC
:18032D001378FE20CA0403EBC3BE021BEBC3BB02EB23C3BB02CD0301EB
:18034500CD2301FE222808FE7F28131213 18EECD0301CD2301FE20CAD2
:18035D000403EBC3BB020E08CD06010E20CD06010E08CD06011B18CD40
:0000000000
```

Abb. 2.1.3-2 Object Code im TDL-Relocating Format

Dort muß das eingegebene Zeichen stehen. C0 bewirkt die Ausgabe eines Zeichens auf der Konsole, wobei das Zeichen dazu im Register C enthalten ist. Es muß nach Aufruf auch im Akku stehen. Die anderen Register dürfen nicht verändert werden. CSTS überprüft den Zustand der Konsole. Ist ein Zeichen eingegeben worden, so wird ØFFH im Akku übergeben, andernfalls Ø. EXIT ist ein Sprung zu der Stelle, wo gegebenenfalls ein anderes Monitorprogramm steht. Der Sprung kann auch zum Monitor selbst geschaltet werden.

Abb. 2.1.3-2 zeigt den Objekt-Code im TDL Relocating Format. Das Programm kann z.B. mit dem in einem folgenden Kapitel beschriebenen Relocator geladen werden, falls ein anderer Monitor schon vorhanden ist. *Abb. 2.1.3-3* zeigt den auf die Adresse 1ØØH verschobenen Monitor im HEX Listing Format. Bei RESET startet der Prozessor normalerweise das Programm auf der Adresse Ø. Da es aber sehr unpraktisch ist, an diese Stelle ein Monitorprogramm zu setzen, weil dort die vielfältige CPM Software läuft, ist es recht günstig, den Monitor z.B. auf die Adresse FØØØ zu legen. Damit der Prozessor trotzdem nach dem RESET dort anfängt das Programm auszuführen und nicht von Ø, ist eine spezielle POWER ON RESET-Logik nötig, die den Prozessor an die entsprechende Stelle bringt (vgl. Abschnitt 1).

```
0100 C3 29 02 C3 03 F0 C3 09 F0 C3 12 F0 C3 1E F0 F5 .).............
0110 1F 1F 1F 1F CD 18 01 F1 E6 0F C6 30 FE 3A DA 23 ...........0.:.#
0120 01 C6 07 4F C3 06 01 7C CD 0F 01 7D 18 E1 21 00 ...O...\...}..!.
0130 00 CD 03 01 CD 23 01 CD 45 01 DA 55 01 29 29 29 .....#..E..U.)))
0140 29 B5 6F 18 EC D6 30 D8 FE 17 3F D8 FE 0A 3F D0 ).O...0...?...?.
0150 D6 07 FE 0A C9 79 FE 0D CA 5C 01 C9 F5 0E 0A CD .....Y...\......
0160 06 01 F1 4F C9 0E 0D CD 06 01 0E 0A 18 98 7E 23 ...O..........~#
0170 B7 C8 CD 23 01 18 F7 06 00 E5 D1 7E E6 DF FE DD ...#.......~....
0180 CA ED 01 7E FE CB CA E7 01 FE ED CA DB 01 7E FE ...~..........~.
0190 C3 CA 24 02 FE CD CA 24 02 E6 EF FE 22 CA 24 02 ..$....$....".$.
01A0 FE 2A CA 24 02 E6 CF FE 01 CA 24 02 E6 C7 FE C2 .*.$......$.....
01B0 CA 24 02 FE C4 CA 24 02 7E E6 F7 FE 10 CA 25 02 .$....$.~.....%.
01C0 FE D3 CA 25 02 E6 E7 FE 20 CA 25 02 E6 C7 FE 06 ...%.... .%.....
01D0 CA 25 02 FE C6 CA 25 02 C3 26 02 23 7E E6 C7 FE .%....%..&.#~...
01E0 43 CA 23 02 C3 25 02 C3 25 02 C3 23 02 23 7E FE C.#..%..%..#.#~.
01F0 CB CA EA 01 FE 21 CA 23 02 E6 FE FE 34 CA 24 02 .....!.#....4.$.
0200 E6 F8 FE 70 CA 24 02 7E E6 CF FE 06 CA 23 02 E6 ...P.$.~.....#..
0210 C7 FE 02 CA 23 02 7E D6 40 E6 87 FE 06 CA 24 02 ....#.~.@.....$.
0220 C3 25 02 04 04 04 04 EB C9 CD 64 02 CD 65 01 21 .%........D..E.!
0230 37 02 CD 6E 01 18 14 52 44 4B 20 4D 4F 4E 49 54 7..N...RDK MONIT
0240 4F 52 20 56 31 2E 30 0D 0A 3E 00 CD 03 01 4F CD OR V1.0..>....O.
0250 06 01 FE 45 CA B8 02 FE 47 CA 85 02 FE 51 CA 89 ...E....G....Q..
0260 02 C3 2C 02 C9 0E 0D CD 06 01 0E 0A 18 98 7E 23 ..,...........~#
0270 B7 C8 CD 23 01 18 F7 06 00 E5 D1 7E E6 DF FE DD ...#.......~....
0280 CA ED 01 7E FE CD 2E 01 E9 CD 03 01 CD 23 01 FE ...~.........#..
0290 49 28 14 CD 2E 01 4D FE 20 C2 2C 02 C5 CD 2E 01 I(....M. .,.....
02A0 C1 7D ED 79 C3 2C 02 CD 2E 01 0E 20 CD 06 01 4D .}.y.,..... ...M
02B0 ED 78 CD 0F 01 C3 2C 02 CD 2E 01 CD 65 01 CD 27 .x....,.....E..'
02C0 01 CD 77 01 C5 E5 0E 20 CD 06 01 7E CD 0F 01 23 ..w.... ...~...#
02D0 10 F4 D1 C1 3E 04 90 47 07 E6 0F 80 47 0E 20 CD ....>..G....G. .
02E0 06 01 10 F9 0E 3A CD 06 01 1A E6 7F FE 20 DA FA .....:..... ....
02F0 02 FE 7F CA FA 02 4F C3 FC 02 0E 20 CD 06 01 0E ......O.... ....
0300 3A CD 06 01 CD 03 01 4F CD 06 01 FE 7F CA 38 03 :.......O......8.
0310 FE 0D CA 3D 03 FE 20 CA BB 02 FE 2E CA 2C 02 FE ...=.. ......,..
0320 22 CA 42 03 21 00 00 CD 37 01 47 7D 12 13 78 FE ".B.!...7.G}..X.
0330 20 CA 04 03 EB C3 BE 02 1B EB C3 BB 02 EB 23 C3  .............#.
0340 BB 02 CD 03 01 CD 23 01 FE 22 28 08 FE 7F 28 13 ......#..".(...(.
0350 12 13 18 EE CD 03 01 CD 23 01 FE 20 CA 04 03 EB ........#.. ....
0360 C3 BE 02 0E 08 CD 06 01 0E 20 CD 06 01 0E 08 CD ......... ......
0370 06 01 1B 18 CD 00 00 00 00 00 00 00 00 00 00 00 ................
-^C
```

Abb. 2.1.3-3 Monitor auf Startadresse 100H

2.1.4 Beispiel eines kommerziellen Monitorprogramms

Hier wird noch auf den TDL–Monitor eingegangen. Der von der Firma TECHNICAL DESIGN LABs mit Listing erhältlich war.

Der Monitor startet bei der Adresse FØØØ und hat eine Kapazität von 2K Byte. Er arbeitet mit I0-Kanälen der S100-Karte SMB 2. Die Kanalnummern können aber leicht auf andere Systeme übertragen werden.

Der Monitor startet mit einer Sprungtabelle für die verschiedenen Peripheriegeräte:

JMP Start	Monitor Startadresse
JMP CI	Konsole Eingabe eines Zeichens
JMP RI	Eingabe eines Zeichens vom Leser

JMP C0	Ausgabe eines Zeichens auf der Konsole
JMP P0	Ausgabe eines Zeichens auf dem Stanzer
JMP L0	Ausgabe eines Zeichens auf dem Drucker
JMP CSTS	Überwachung des Status der Konsole
JMP I0BYTE	I/0 Kanal Zuweisung, Abfrage (hier kurzgeschlossen)-
JMP I0SET	I/0 Kanäle zuweisen
JMP MEMCHK	Höchster Speicherplatz in B und A
JMP RESTART	Wiedereintritt im Falle eines BREAKPOINT

Auf der Adresse FØ23 befindet sich die Initialisierung der I0-Ports. Port 7ØH ist der TTY-Port, Port 72H der CRT-Port und Port 74H der Cassetten-Interface-Port. Die einzelnen Routinen sind anhand der Adressen in der Sprungtabelle leicht auffindbar.
Abb. 2.1.4-1 zeigt das Object Listing des Monitors.

Auf der Adresse F8ØØH, auf der z.B. RAM liegen kann, befindet sich eine andere Sprungtabelle. Diese kann vom Benutzer eingetragen werden.

JMP CIL0C	Konsol-Eingabe von der Benutzer-Konsole
JMP C0L0C	Konsol-Ausgabe vom Benutzer
JMP RPTPL	Schnell-Leser vom Benutzer
JMP RUL0C	Leser vom Benutzer
JMP PTPL	Schnellstanzer vom Benutzer
JMP PUL0C	Stanzer des Benutzers
JMP LÜL0C	Drucker vom Benutzer (Zeilendrucker und weitere Drucker)
JMP CSL0C	Konsol Status vom Benutzer

Um den Monitor in Betrieb nehmen zu können, müssen zunächst nur die Initialisierung der Konsole und die Routinen CI und C0 geschrieben werden.

Eine Besonderheit stellt der Port 76H dar. Er dient der Zwischenspeicherung der I0-Zuweisung. Damit wird bestimmt, welchem physikalischen Gerät eine bestimmte Logik zugeordnet wird (vgl. Befehl A). Bei der Inbetriebnahme muß darauf geachtet werden, daß die Eingabe von diesem Gerät immer Kanal Ø ergibt. Dann können die Routine CI auf die Adresse F61FH, die Routine C0 auf die Adresse F49ØH und die Routine CSTS auf die Adresse F52ØH geschrieben werden. Die nachstehende Liste gibt eine Zusammenstellung der Befehle.

A
Zuweisung der physikalischen Geräte zu den Logik-Angaben. Logiken sind: Konsole (C), READER (R), PUNCH (P) LIST DEVICE (L). Als sogenannte physikalische Geräte werden genannt: Bei der Konsole TTY (T), VIDEO (V), BATCH (B), USER (U), beim Reader TTY (T), CASSETTE (C), PAPER (P), Benutzer-Routine USER (U). Beim Punch TTY (T), CASSETTE (C), PAPER (P), USER (U). Bei der Listdevice TTY (T), VIDEO (V), LINE PRINTER (L), Benutzer Routine USER (U). Wird vom Port 76H (I0STATUS) der Wert 0 gelesen, so gelten automatisch die Zuweisungen: AC=T, AR=T, AP=T und AL=T. Das heißt, alle logischen Geräte sind auf das physikalische Gerät TTY (Konsol Routine) geschaltet. Bei der Anweisung AC=B wird die Ausgabe C0 auf den Drucker (L0) geschaltet und die Eingabe kommt von R1.

```
>DF000 F7FF

F000  C3 32 F0 C3 19 F6 C3 36 F6 C3 8A F4 C3 C4 F4 C3
F010  AB F4 C3 1A F5 DB 76 C9 C3 1D F1 C3 AC F5 C3 BE
F020  F6 0D 0A 00 00 00 5A 61 70 70 6C 65 20 56 31 2E
F030  31 52 3E 53 D3 70 D3 72 D3 74 3E 51 D3 70 D3 72
F040  3D D3 74 AF D3 77 D3 7A 3D D3 76 3E 04 D3 77 DB
F050  7A D3 76 3E F8 ED 47 31 5B F0 C3 BA F5 5F F0 F9
F060  EB 01 23 00 21 A8 F7 ED B0 EB 01 A1 FF 09 E5 21
F070  00 00 06 0A E5 10 FD 06 11 CD 4F F4 11 7C F0 D5
F080  CD 12 F5 0E 3E CD 8A F4 CD 36 F7 E6 7F 28 F9 D6
F090  41 F8 FE 1A D0 87 21 A2 F0 85 6F 7E 23 66 6F 0E
F0A0  02 E9 D6 F0 21 F1 4E F1 6F F1 86 F1 A2 F1 AF F1
F0B0  7E F5 1B F8 FD F1 1E F8 81 F6 1B F2 F8 F4 21 F8
F0C0  2F F1 57 F7 26 F2 DF F2 08 F3 E0 F4 82 F7 70 F3
F0D0  B0 F3 28 F3 7B F4 CD 36 F7 21 94 F7 01 00 04 11
F0E0  05 00 BE 28 06 19 0C 10 F9 18 15 59 CD 36 F7 FE
F0F0  3D 20 F9 CD 36 F7 01 00 04 23 BE 28 06 0C 10 F9
F100  C3 64 F4 3E 03 1C 1D 28 08 CB 21 CB 21 17 17 18
F110  F5 2F 57 CD 0A F6 30 FB DB 76 A2 B1 4F 79 D3 76
F120  C9 CD 12 F5 CD 30 F7 FE 1E 20 F9 D1 C3 77 F0 CD
F130  40 F5 CD 12 F5 E1 CD 30 F7 FE 04 CA 82 F4 FE 5F
F140  28 08 77 4F 23 CD 8A F4 18 EC 2B 4E 18 F7 CD 0D
F150  F5 CD 74 F4 BE C4 5D F1 CD 6E F5 18 F4 47 CD 85
F160  F4 7E CD 8F F5 CD 88 F4 78 CD 8F F5 C3 12 F5 CD
F170  0D F5 CD 82 F4 CD 88 F4 7E CD 8F F5 CD 6E F5 7D
F180  E6 0F 20 F1 18 EC CD 40 F5 CD 8D F4 0E 3A CD C4
F190  F4 AF CD EE F5 E1 CD E9 F5 21 00 00 CD E9 F5 C3
F1A0  F8 F4 CD 35 F5 71 CD 74 F5 30 FA D1 C3 7C F0 CD
F1B0  0A F6 38 40 28 10 CD 67 F5 D1 21 34 00 39 72 2B
F1C0  73 78 FE 0D 28 2E 16 02 21 35 00 39 E5 CD 40 F5
F1D0  58 C1 E1 78 B1 28 0A 71 23 70 23 0A 77 23 3E FF
F1E0  02 7B FE 0D 28 03 15 20 E3 3E C3 32 38 00 21 1E
F1F0  F0 22 39 00 CD 12 F5 D1 21 16 00 39 E9 CD 0D F5
F200  7E 47 2F 77 AE 28 0E D5 50 5F CD 85 F4 CD 69 F7
F210  CD 12 F5 42 D1 70 CD 6E F5 18 E5 CD 35 F5 7E 02
F220  03 CD 6E F5 18 F8 CD 40 F5 78 D6 0D 47 4F D1 28
F230  04 CD 40 F5 C1 EB D9 CD 12 F5 CD 7B F6 D6 3A 47
F240  E6 FE 20 F6 57 CD C0 F2 5F CD C0 F2 F5 CD C0 F2
F250  D9 D1 5F C5 D5 E5 19 E3 DD E1 D9 E1 CD C0 F2 3D
F260  78 C1 20 03 09 DD 09 1C 1D 28 19 3D 28 22 CD C0
F270  F2 CD D3 F2 20 F8 CD C0 F2 28 BF DD E5 E1 CD 8A
F280  F5 C3 64 F4 7C B5 C8 EB 21 34 00 39 72 2B 73 C9
F290  2E 01 CD B0 F2 38 07 CD D3 F2 20 F6 18 D8 4F CD
F2A0  B0 F2 47 D9 C5 D9 E3 09 7D CD D3 F2 7C E1 18 E7
F2B0  2D 20 07 CD C0 F2 1D 67 2E 08 CD C0 F2 CB 24 C9
F2C0  C5 CD D6 F5 07 07 07 07 4F CD D6 F5 B1 4F 82 57
F2D0  79 C1 C9 DD 77 00 DD BE 00 20 A0 DD 23 1D C9 CD
F2E0  40 F5 E1 7E CD 8F F5 CD 05 F6 D8 28 0F FE 5F 28
F2F0  14 E5 CD 67 F5 D1 E1 73 78 FE 0D C8 23 7D E6 07
F300  CC 82 F4 18 DE 2B 18 F5 CD 0D F5 CD 82 F4 06 30
F310  7E E6 7F FE 20 30 02 3E 2E FE 7C 30 FA 4F CD 8A
F320  F4 CD 6E F5 10 EA 18 E3 16 00 CD 40 F5 E1 65 E5
F330  33 14 78 D6 0D 20 F3 47 4F 67 6A 2D 39 E5 C5 C5
F340  CD 12 F5 C1 E1 DD E1 5A DD 7E 00 ED B1 E2 4B F3
F350  DD E5 E5 1D 28 0B DD 7E FF BF 20 E8 23 DD 2B 18
F360  F2 E1 E5 2B C5 CD 8A F5 C1 18 D4 33 1D 20 FC C9
F370  CD 0D F5 CD F8 F4 CD 8D F4 01 3A 00 CD C4 F4 D5
F380  E5 04 CD 74 F5 38 24 3E 18 90 20 F5 E1 CD 93 F3
F390  D1 18 E3 57 78 CD EE F5 CD E9 F5 AF CD EE F5 7E
F3A0  CD EE F5 23 10 F9 AF 92 C3 EE F5 E1 D1 AF 18 E3
```

Abb. 2.1.4-1 Object Code des TDL-Monitors

```
F3B0  CD 36 F7 21 CB F7 FE 0D 28 5A FE 27 20 0A 21 E7
F3C0  F7 CD 36 F7 FE 0D 28 4C BE 28 09 CB 7E C2 64 F4
F3D0  23 23 18 F4 CD 88 F4 23 7E 47 E6 3F EB 6F 26 00
F3E0  39 EB 23 1A CD 8F F5 CB 78 28 05 1B 1A CD 8F F5
F3F0  CD 05 F6 D8 28 19 E5 C5 CD 67 F5 E1 F1 C5 F5 7D
F400  12 C1 CB 78 28 03 13 7C 12 C1 E1 78 FE 0D C8 CB
F410  7E C0 18 C3 CD 12 F5 CD 88 F4 7E 23 B7 F8 4F CD
F420  8A F4 0E 3D CD 8A F4 7E 47 E6 3F 23 EB 6F 26 00
F430  39 EB CB 70 20 0F 1A CD 8F F5 CB 78 28 D9 18 1A
F440  CD 8F F5 18 D2 E5 1A 67 18 1A 6F 7E E1 18 F1 21
F450  21 F0 4E 23 CD 8A F4 10 F9 CD 1A F5 B7 C8 CD 30
F460  F7 FE 03 C0 CD B9 F5 11 EA FF 19 F9 0E 2A CD 8A
F470  F4 C3 7C F0 CD 36 F6 38 EB 8A C9 CD B9 F5 01 23
F480  00 09 CD 12 F5 CD 8A F5 0E 20 DB 76 E6 03 20 0A
F490  DB 70 E6 02 28 FA 79 D3 71 C9 3D 20 0A DB 72 E6
F4A0  02 28 FA 79 D3 73 C9 3D C2 03 F8 DB 76 E6 C0 28
F4B0  DF FE 40 28 EB FE 80 CA 12 F8 C3 15 F8 0E 0D CD
F4C0  C4 F4 0E 0A DB 76 E6 30 28 C6 FE 20 20 0A DB 74
F4D0  E6 02 28 FA 79 D3 75 C9 FE 10 CA 0C F8 C3 0F F8
F4E0  CD 0D F5 CD EB F4 CD A3 F5 CD 9E F5 4E CD C4 F4
F4F0  CD 74 F5 30 F7 CD 9E F5 CD A3 F5 DB 76 E6 33 C0
F500  C3 88 F0 E6 0F C6 90 27 CE 40 27 4F C9 CD 42 F5
F510  D1 E1 F5 06 05 CD 4F F4 E1 C9 DB 76 E6 03 20 04
F520  DB 70 18 05 3D 20 09 DB 72 E6 01 3E FF C0 2F C9
F530  3D C8 C3 18 F8 0C CD 42 F5 CD 12 F5 C1 D1 E1 C9
F540  0E 01 21 00 00 CD 36 F7 47 CD D9 F5 38 08 29 29
F550  29 29 B5 6F 18 EF E3 E5 78 CD 0D F6 30 02 0D C8
F560  C2 64 F4 0D 20 DC C9 0E 01 21 00 00 18 DA CD 74
F570  F5 D0 D1 C9 23 7C B5 37 C8 7B 95 7A 9C C9 CD 0D
F580  F5 E5 19 CD B5 F4 E1 B7 ED 52 7C CD 8F F5 7D F5
F590  0F 0F 0F 0F CD 98 F5 F1 CD 03 F5 C3 8A F4 01 FF
F5A0  08 18 03 01 00 48 CD C4 F4 10 FB C9 E5 CD B9 F5
F5B0  7D D6 3C 30 01 25 44 E1 C9 C5 21 FF FF 24 7E 2F
F5C0  77 BE 2F 77 20 F7 24 7E 2F 77 BE 2F 77 28 F7 25
F5D0  01 DD FF 09 C1 C9 CD 7B F6 D6 30 D8 FE 17 3F D8
F5E0  FE 0A 3F D0 D6 07 FE 0A C9 7C CD EE F5 7D F5 0F
F5F0  0F 0F 0F CD 03 F5 CD C4 F4 F1 F5 CD 03 F5 CD C4
F600  F4 F1 82 57 C9 0E 2D CD 8A F4 CD 36 F7 FE 20 C8
F610  FE 2C C8 FE 0D 37 C8 3F C9 DB 76 E6 03 20 08 DB
F620  70 1F 30 FB DB 71 C9 3D 20 08 DB 72 1F 30 FB DB
F630  73 C9 3D C2 00 F8 E5 DB 76 E6 0C 2F D3 7A 2F D3
F640  7A 20 1A 67 DB 70 1F 38 0F C5 06 00 E3 E3 10 FC
F650  C1 25 20 F0 AF 37 E1 C9 DB 71 B7 E1 C9 FE 08 20
F660  11 DB FF 6F DB FF 8D 20 EB DB 74 1F 30 F6 DB 75
F670  18 E8 E1 FE 04 CA 06 F8 C3 09 F8 CD 74 F4 E6 7F
F680  C9 CD 40 F5 E1 CD 12 F5 16 FF 06 04 CD 74 F4 20
F690  F9 10 F9 CD 74 F4 28 FB 77 3E 07 D3 71 23 CD 74
F6A0  F4 28 03 77 18 F7 1E 01 CD 74 F4 20 09 1C 3E 07
F6B0  BB 20 F5 C3 8A F5 72 23 1D 20 FB 77 18 DF E5 D5
F6C0  C5 F5 CD B9 F5 EB 21 0A 00 39 06 04 EB 2B 72 2B
F6D0  73 D1 10 F9 C1 0B F9 21 25 00 39 7E 91 23 20 04
F6E0  7E 90 28 0C 23 23 7E 91 20 05 23 7E 90 28 01 03
F6F0  21 20 00 39 73 23 72 23 23 71 23 70 C5 0E 40 CD
F700  8A F4 E1 CD 8A F5 21 25 00 39 01 00 02 5E 71 23
F710  56 71 23 7B B2 28 02 7E 12 23 10 F1 08 D9 E5 D5
F720  C5 F5 DD E5 FD E5 ED 57 47 ED 5F 4F C5 C3 7C F0
F730  CD 19 F6 E6 7F C9 CD 30 F7 C8 3C F8 3D FE 0D C8
F740  FE 4E C8 FE 6E 28 0D C5 4F CD 8A F4 79 C1 FE 40
F750  D8 FE 7B D0 E6 5F C9 CD 36 F7 FE 4F 28 1C FE 49
F760  C2 64 F4 CD 40 F5 C1 ED 58 46 08 CD 88 F4 CB 23
F770  3F 18 8F 4F CD 8A F4 10 F5 C9 CD 42 F5 D1 C1 ED
F780  59 C9 CD 35 F5 0A BE 28 05 C5 CD 5D F1 C1 03 CD
```

zu Abb. 2.1.4-1

```
F790  6F F5 18 F1 43 54 56 42 55 52 54 50 43 55 50 54
F7A0  50 43 55 4C 54 56 4C 55 C1 79 ED 4F 78 ED 47 FD
F7B0  E1 DD E1 F1 C1 D1 E1 08 D9 D1 C1 F1 E1 F9 00 21
F7C0  00 00 C3 00 00 00 00 00 00 00 00 41 15 42 13 43
F7D0  12 44 11 45 10 46 14 48 31 4C 30 4D F1 50 B4 53
F7E0  97 49 03 F5 CA 3F C7 41 09 42 0B 43 0A 44 0D 45
F7F0  0C 46 08 48 0F 4C 0E 4D CF 58 87 59 85 52 02 C1
```

zu Abb. 2.1.4-1

B
Der Befehl bewirkt eine Stillegung der Eingabetastatur. Der Monitor meldet sich erst wieder bei der Eingabe von RS (Code 1EH, CTRL N).

C
Vergleichen der Lesereingabe mit dem Speicher. Das Format lautet Canfadr, endadr er.

D
Anzeigen eines Speicherbereichs. Danfadr, endadr er.

E
Dieser Befehl erzeugt das End of file Zeichen für das TDL und INTEL HEX Format, das dazu auf dem PUNCH ausgegeben wird. Dabei kann eine zusätzliche Adresse angegeben werden, die dann von der Lade-Routine als Anfangsadresse interpretiert wird. Das Format lautet Eadr cr.

F
Ein Speicherbereich wird mit einer Konstanten gefüllt. Dies kann zum Beispiel bei der EPROM-Programmierung erforderlich sein. Fanfadr, endadr, data füllt den Bereich von anfadr bis endadr mit dem Wort data.

G
GOTO Befehl. Gadr startet ein Programm an der Adresse adr. Mit Gadr, br1, br2 cr kann ein Programm auch mit BREAK Points gestartet werden, die bewirken, daß der Monitor wieder aufgerufen wird, wenn die angegebene Stelle erreicht wird. Dann können z.B. die Registerinhalte angesehen werden. Dann kann das Programm an der Stelle mit Gcr wieder gestartet werden und weiterlaufen. BREAK Points dürfen nur auf Anfänge von Befehlscodes gesetzt werden.

H
Hdata1, data2cr berechnet die Summe und die Differenz der beiden Werte data1 und data2.

J
Hier kann ein Speicherbetrieb grob getestet werden. Dazu hat J das Format Janfadr, endadrcr. Der Test wirkt nicht störend auf den bestehenden Speicherinhalt an dieser Stelle.

L
Laden einer Binärfile. Mit Ladrcr wird eine Binärfile vom READER geladen, beginnend auf der Adresse adr. Die Routine paßt direkt zu dem Befehl U. Der Anfang der Binärfile wird durch eine Anzahl von 8FFH eingeleitet.

M
Transport-Befehl. Mit Manfadr, endadr, zieladr cr wird ein Datenblock von der anfadr zur endadr nach zieladr verschoben. Die Blöcke dürfen sich dabei nicht überlappen.

N
Ausgabe von NULL-Zeichen an den PUNCH. Es werden 72 Byte Code Ø ausgegeben. Dieser Befehl ist zur Synchronisierung der PE-Cassetten-Schaltung erforderlich und muß daher vor jeder Aufzeichnung von Daten bei diesem Interface angewendet werden.Bei Lochstreifen-Geräten dient er dem Lochstreifen-Vorschub.

P
Eingeben von 7-Bit-Zeichen (ASCII) in den Speicher. Das Format lautet Padr cr. Mit CTRL D kann dieser Eingabemodus wieder verlassen werden. Mit backspace (hier das Zeichen „_" kann ein versehentlich eingegebenes Zeichen wieder gelöscht werden.

Q
Dieser Befehl wirkt genau so, wie beim im Abschnitt 2.1.2 beschriebenen Monitor. Qlportadr cr: Eingabe eines Wortes, Q0portadr data cr: Ausgabe eines Wertes an einen Port.

R
Einlesen einer HEX File. Diese File enthält CHECKSUM-Zeichen zum Überwachen des Einlesevorgangs. Es werden zwei Arten unterschieden: INTEL HEX File, sie ist absolut, sowie TDL RELOCATING File mit Informationen über Sprungadressen und Speicheradreßbezügen. Damit ist es möglich, ein Programm an eine beliebige Stelle im Speicher lauffähig zu laden.

Rcr
Es wird eine HEX File an die Adresse im Speicher gelesen, die in der File vorgegeben ist.

Rbias cr
Eine File wird an die in der File angegebenen Adresse und bias geladen. Dort ist die File im allgemeinen nicht startbar.

R, reladrcr
Eine File wird an der Stelle reladr geladen, falls es sich um eine im TDL RELOCATING Format abgelegte File handelt. Sie ist dort direkt ausführbar.

Rbias, reladr cr
Hier wird eine File an die Stelle reladr und bias geladen. Das gilt nur für RELOCATING Format.

S
Modifizieren von Speicherinhalten. Sadr sp, wobei sp für space bzw. blank gesetzt wird, es wird der Inhalt der Adresse adr ausgegeben. Mit cr wird der Befehl abgeschlossen. Wird blank eingegeben, so wird die nächste Zelle ausgegeben. Mit backstep (hier „_" Code 5FH) wird die vorhergehende Zelle ausgegeben. Bei Eingabe eines Sedezimalwertes wird dieser anstelle des vorher angezeigten Wertes in die Speicherzelle abgelegt.

T
Tanfadr, endadr cr
Hier werden die Angaben im Bereich anfadr bis endadr ausgedruckt. Dabei werden alle darstellbaren Zeichen ausgegeben und die nicht darstellbaren Zeichen durch einen Punkt ersetzt.

U
Binärausgabe. Uanfadr, endadr cr gibt einen Speicherbereich an den PUNCH.

V
Vanfadr, endadr, zieladr cr. Es wird der Bereich von anfadr bis endadr mit zieladr verglichen. Unterschiede werden ausgegeben.

W
Wanfadr, endadrcr. Es wird ein Speicherbereich im INTEL HEX Format ausgegeben. Am Schluß muß mit Ecr das Endezeichen zusätzlich an den PUNCH gegeben werden.

X
Registerinhalte ausgeben. Xcr gibt die Inhalte der Register A, B, C, D, E, F, H, L, M, P, S, I aus. X'cr gibt die Inhalte der Register A', B', C', D', E', F', H', L', M, X, Y, R aus. Mit XA kann z.B. der Inhalt des Akkus modifiziert werden.

Y
Suchen nach mehreren Datenbytes. Ydata1, data2, data3 . . . cr sucht im gesamten Speicher nach dem Datenmuster. Maximal können 255 Datenbytes angegeben werden. Mit CTRL C läßt sich der Suchvorgang beenden.

Z
Zcr gibt den höchsten Speicherplatz aus. Er liegt am Ende des ersten zusammenhängenden Speicherblocks.

2.2 Editor

Für einen Assembler ist es erforderlich, einen einzugebenden Text zuvor aufzubereiten. Die primäre Aufgabe wäre es also, den Text erst einmal in einen Speicher einzugeben. Dies könnte mit dem Befehl Padr cr (siehe Abschnitt 2.1) geschehen. Soll aber nachträglich ein Wort mitten in den Text eingefügt werden, so beginnen bereits die Schwierigkeiten. Es müßte dann der restliche Text von der betreffenden Stelle an um die Anzahl der Bytes verschoben werden, die der neue Text einnimmt. Da dies mit einem Monitor sehr umständlich und fehlerträchtig ist, wurden Programme geschaffen, die das ermöglichen. Man nennt sie EDITOR, sie dienen der Eingabe von Texten, deren Aufbereitung und Verbesserung. Im folgenden wird ein kleiner EDITOR beschrieben, der etwa 1K Byte benötigt.

2.2.1 Befehle des EDITORS

Nach Start des EDITORs meldet sich dieser mit dem Zeichen *. Es können dann verschiedene Befehle zur Steuerung des Eingabevorganges gegeben werden. *Abb. 2.2.1-1* zeigt ein Beispiel.

I
Eingabe eines Textes in den Textspeicher. Alle nach I folgenden Zeichen werden im Speicher an der aktuellen Buffer-Adresse abgelegt. Mit rubout (7FH) oder backspace (Ø8H) kann ein falsch eingegebenes Zeichen gelöscht werden. Dabei wandert der Cursor nach links. Liegt ein Zeilenvorschub vor, so wird auf die vorhergehende Zeile zurückgeschaltet. Die Texteingabe wird mit CTRL Z (1AH) abgeschlossen.

```
*

*I
EINFUEGEN EINES TESTTEXTES
A
B
C
D
ENDE
```

Abb. 2.2.1-1 Beispiel einer EDITOR-Bedienung

```
*B
*10T
EINFUEGEN EINES TESTTEXTES
A
B
C
D
ENDE

*2L
*T A

*T A

*2T A
B

*K
*B
*10T
EINFUEGEN EINES TESTTEXTES
B
C
D
ENDE

* EINFUEGEN EINES TESTTEXTES
* B
*IHINZUFUEGEN EINES WEITEREN TEXTES

*B
*20T
EINFUEGEN EINES TESTTEXTES
HINZUFUEGEN EINES WEITEREN TEXTES
B
C
D
ENDE

*Z
*-10T
EINFUEGEN EINES TESTTEXTES
HINZUFUEGEN EINES WEITEREN TEXTES
B
C
D
ENDE
```

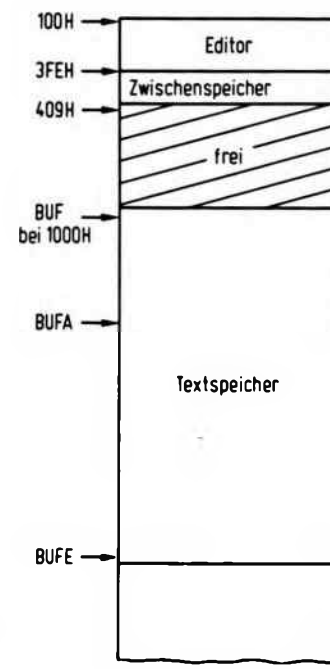


Abb. 2.2.2-1 Speicherverteilung bei EDITOR (Texts. 124)

T
anzahlT (wobei statt anzahl die Zahl der Zeilen bestimmt) druckt die nächsten Zeilen, die im Textbuffer stehen, aus. -anzahlT druckt die Anzahl der davor stehenden Zeilen aus.

B
Hierdurch wird der aktuelle Bufferzeiger wieder auf den Anfang gestellt.

Z
Hierdurch wird der Bufferzeiger an den Schluß des Textbufferbereichs.

L
anzahlL (für anzahl siehe oben) schiebt den aktuellen Bufferzeiger um die entsprechende Anzahl Zeilen vor.
-anzahlL schiebt den Bufferzeiger um die Anzahl der Zeilen zurück. Beide Befehle können nicht über die Grenzen des Bufferanfangs und des Bufferendes ausgeführt werden.

K
Löschen einer Zeile. Es wird die durch den aktuellen Bufferzeiger bestimmte Zeile gelöscht. Mit T kann die Zeile zur Kontrolle vor Anwendung dieses Befehls ausgegeben werden.
space
Die nächste Zeile wird ausgegeben, und der Bufferzeiger wird um eine Zeile erhöht. Dieses Zeichen hat dieselbe Wirkung wie die Zeichenfolge LT.

E
Ende des Editierens. Das Monitorprogramm wird wieder gestartet. Ein erneuter Rücksprung in den EDITOR ist durch Sprung auf die Adresse RSTART möglich. Ein Sprung auf den Anfang des EDITORs würde den vorher editierten Text unzugänglich machen.

2.2.2 Funktionsweise des EDITORS

Abb. 2.2.2-1 zeigt die Speicherverteilung, wie sie vom EDITOR angelegt wird. Der EDITOR selbst befindet sich auf der Adresse 1ØØH und reicht bis zur Adresse 3FEH. Auf dieser Adresse beginnt ein Zwischenspeicherbereich, der bis 4Ø9H geht. Der eigentliche Textspeicher startet bei der Adresse 1ØØØH und reicht bis zu einer Stelle, die mit BUFE bezeichnet ist. Das Ende des Textspeichers kann nicht angegeben werden, da es von dem im Textspeicher befindlichen Text abhängt. Am Anfang zeigen BUFE und BUFA auf die Stelle BUF. Das heißt, es befindet sich kein Text in dem Speicher. Nur mit dem Befehl I kann erreicht werden, daß BUFE erhöht wird. BUFA zeigt immer auf die aktuelle Zeile. Diese Zeile wird z.B. durch den Befehl T angezeigt. Durch den Befehl K kann die betreffende Zeile gelöscht werden. Nach einem Löschvorgang wird der Zeiger BUFE wieder erniedrigt, ebenfalls nach Eingabe von rubout oder backspace im Befehl I. Der maximal editierbare Text hängt von der Größe des vorhandenen Speichers in dem Computersystem ab.

Abb. 2.2.2-2 zeigt das grobe Flußdiagramm des EDITORS. Nach dem Starten des EDITORS und entsprechender Vorbesetzung der Zellen BUFA und BUFE kommt das Programm in eine große Schleife. Dort wird zunächst ein Zeichen von der Konsole eingelesen, dann wird entschieden zu welchem Teilprogramm gesprungen wird, um einen eventuell gegebenen Befehl auszuführen. E bewirkt einen direkten Sprung in das Monitorprogramm. B, Z beeinflussen z.B. den Zwischenspeicher BUFA entsprechend. cr und lf

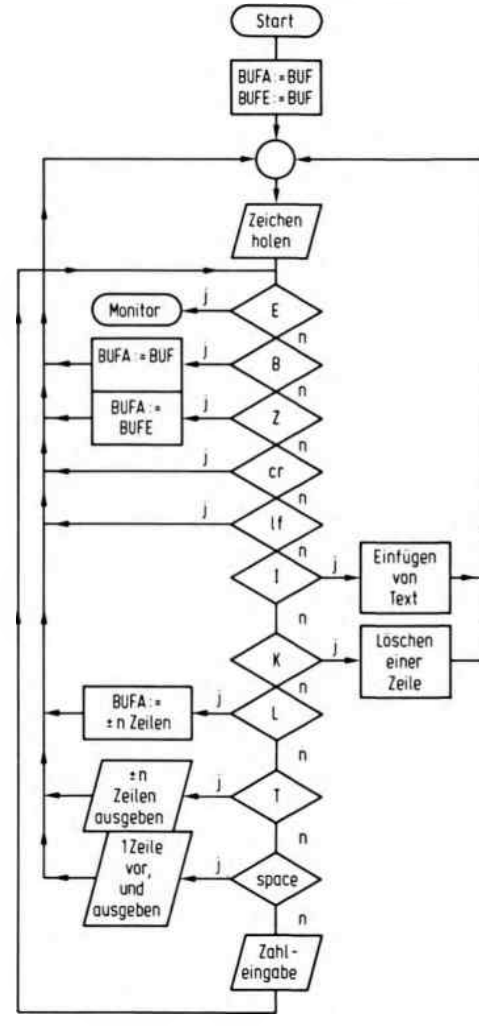


Abb. 2.2.2-2 Flußdiagramm des EDITORS

bewirken nichts. I geht zu einem umfangreicheren Teilprogramm INSERT. K löscht eine Zeile und gelangt dazu ebenfalls in ein größeres Teilprogramm. L erhöht BUFA um die entsprechende Anzahl von Bytes, die sich aus der Länge einer Zeile und der Anzahl der Zeilen ergibt. T muß ebenfalls die Länge der Zeilen kennen, um den Text richtig formatiert ausgeben zu können.

Space ruft im wesentlichen L und T auf, so daß die gleiche Wirkung erzielt wird. In jedem anderen Fall wird zu einer Routine „ZAHLENEINGABE" gesprungen. Diese bewirkt folgendes: Falls es sich bei dem eingegebenen Zeichen um + oder – oder eine Zahl gehandelt hat, wird diese in einen Zwischenspeicher übertragen. Nach Rückkehr aus dem Programm bei erneuter Eingabe eines Wertes, falls dieser keine Zahl war, steht dann den Befehlen L, T der komplette Zahlenwert mit Vorzeichen zur Verfügung.

Abb. 2.2.2-3 zeigt das komplette Listing des EDITORS. Am Anfang steht die nun

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 1
.MAIN. -


                        ;
                        .PHEX
                        .PABS
0100                    .LOC 100H
                        ;
                        ;*********************************
                        ;* A TINY LINE EDITOR            *
                        ;* BY ROLF-D. KLEIN              *
                        ;* V790604                       *
                        ;*********************************
                        ;
                        ;
                        ;
0100    C3 020C         JMP START
0103    C3 0203         JMP ENTRY       ;NEUSTART MIT ALTEN VEKTOREN
0106    C3 F003         CI: JMP 0F003H
0109    C3 F009         CO: JMP 0F009H
010C    C3 F012         CSTC:JMP 0F012H
010F    C3 F00F         LP: JMP 0F00FH
0112    C3 F01E         EXIT: JMP 0F01EH
                        ;
                        ;
                        ; * UNTERPROGRAMME
0115                    PRINT:          ;DRUCKT BIS LF
0115    E5              PUSH H          ;RETTEN POINTER
0116                    PR1:
0116    7C              MOV A,H
0117    BA              CMP D           ;BUFFERENDE
0118    C2 0120         JNZ CONTP
011B    7D              MOV A,L
011C    BB              CMP E
011D    CA 0129         JZ FINPR
0120                    CONTP:
0120    4E              MOV C,M         ;LADEN ASCII WERT
0121    23              INX H
0122    CD 0109         CALL CO
0125    FE0A            CPI 0AH         ;LF AUSGEBEN ?
0127    20ED            JRNZ PR1
0129                    FINPR:
0129    E1              POP H           ;POINTER ZURUECK
012A    C9              RET
                        ;
012B                    PRINTS:         ;DRUCKEN OHNE CRLF
012B    E5              PUSH H
012C                    PR2:
012C    7C              MOV A,H
012D    BA              CMP D           ;ENDE PRUEFEN
012E    C2 0136         JNZ CONPR2
0131    7D              MOV A,L
0132    BB              CMP E
0133    CA 0129         JZ FINPR        ;ROUTINE VON PRINT
0136                    CONPR2:
0136    7E              MOV A,M
0137    FE0A            CPI 0AH
0139    CA 0129         JZ FINPR
```

Abb. 2.2.2-3 Komplette Listing des EDITORs

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                        PAGE 2
.MAIN. - RDK EDITOR V1.0 790604


013C    FE0D        CPI 0DH
013E    CA 0129     JZ FINPR*       ;KEIN CRLF DRUCKEN
0141    23          INX H
0142    4F          MOV C,A
0143    CD 0109     CALL CO         ;AUSGEBEN SONST
0146    18E4        JMPR PR2
                    ;
                    ;
                    ;
0148                LINEP:          ;EINE ZEILE VORSCHUB
0148    7C          MOV A,H
0149    BA          CMP D           ;ENDE BUFFER ERREICHT
014A    C2 0150     JNZ LCON
014D    7D          MOV A,L
014E    BB          CMP E
014F    C8          RZ              ;JA FERTIG NICHT WEITER
0150                LCON:
0150    7E          MOV A,M         ;WERT HOLEN
0151    23          INX H           ;POINTER EINS WEITER
0152    FE0A        CPI 0AH         ;LF
0154    C8          RZ              ;JA DANN FERTIG
0155    C3 0148     JMP LINEP       ;ZURUECK
                    ;
                    ;
0158                LINEM:          ;EINE ZEILE ZURUECK
0158    7C          MOV A,H
0159    FE10        CPI BUFH        ;BUFFERANFANG ?
015B    C2 0162     JNZ LMCON
015E    7D          MOV A,L
015F    FE00        CPI BUFL
0161    C8          RZ              ;JA DANN NICHT WEITER ZURUECK
0162                LMCON:
0162    2B          DCX H           ;POINTER-1
0163                LPMC:
0163    7C          MOV A,H
0164    FE10        CPI BUFH
0166    C2 016D     JNZ COLPM
0169    7D          MOV A,L
016A    FE00        CPI BUFL
016C    C8          RZ              ;BUFFERANFANG
016D                COLPM:
016D    2B          DCX H
016E    7E          MOV A,M         ;SUCHEN NACH WEITEREM LF
016F    FE0A        CPI 0AH
0171    C2 0163     JNZ LPMC
0174    23          INX H           ;EINS WEITER
0175    C9          RET
                    ;
                    ;
0176                KILLEX:         ;EINE ZEILE LOESCHEN
0176    22 03FE     SHLD BUFA       ;RETTEN POINTER
0179                KI1:
0179    7C          MOV A,H         ;BUFFERENDE?
017A    BA          CMP D
017B    C2 0183     JNZ KCO
```

zu Abb. 2.2.2-3

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 3
.MAIN. -- RDK EDITOR V1.0 790604


017E    7D          MOV A,L
017F    BB          CMP E
0180    CA 018A     JZ KK1          ;JA ENDE
0183                KCO:
0183    7E          MOV A,M
0184    23          INX H
0185    FE0A        CPI 0AH         ;LF SUCHEN
0187    C2 0179     JNZ KI1
018A                KK1:
018A    22 0402     SHLD TEMP       ;ENDADRESSE
018D    ED53 0400   SDED BUFE       ;BUFFERENDE
0191    2A 0400     LHLD BUFE       ;BUFE-BUFACTUELL
0194    ED4B 0402   LBCD TEMP
0198    37          STC
0199    3F          CMC
019A    ED42        DSBC B
019C    23          INX H           ;+1
019D    E5          PUSH H
019E    C1          POP B
019F    2A 0402     LHLD TEMP       ;NEUER POINTER
01A2    ED5B 03FE   LDED BUFA       ;ALTER POINTER
01A6    EDB0        LDIR            ;SCHIEBEN
01A8    2A 0402     LHLD TEMP       ;ANZAHL GELOESCHTER ZEICHEN
01AB    ED4B 03FE   LBCD BUFA
01AF    37          STC
01B0    3F          CMC
01B1    ED42        DSBC B
01B3    E5          PUSH H
01B4    D1          POP D
01B5    2A 0400     LHLD BUFE       ;ENDE BUFFER
01B8    37          STC
01B9    3F          CMC
01BA    ED52        DSBC D          ;ENDE BERECHNEN
01BC    22 0400     SHLD BUFE
01BF    E5          PUSH H
01C0    D1          POP D
01C1    2A 03FE     LHLD BUFA       ;POINTER STEHT NOCH
01C4    C9          RET
                    ;
                    ;
                    ;
                    ;
01C5                INP:
01C5    13          INX D           ;BUFE+1
01C6    ED53 0400   SDED BUFE
01CA    22 03FE     SHLD BUFA       ;RETTEN POINTER
01CD    EB          XCHG
01CE    37          STC
01CF    3F          CMC             ;
01D0    ED52        DSBC D          ;LAENGE BERECHNEN
01D2    E5          PUSH H
01D3    C1          POP B
01D4    2A 0400     LHLD BUFE       ;BUFFER ENDE
01D7    E5          PUSH H
01D8    D1          POP D
```

zu Abb. 2.2.2-3

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. - RDK EDITOR V1.0 790604


01D9    2B          DCX H
01DA    EDB8        LDDR            ;SCHIEBEN
01DC    2A 03FE     LHLD BUFA       ;POINTER BLEIBT
01DF    ED5B 0400   LDED BUFE
01E3    C9          RET
                    ;
                    ;
01E4                INM:
01E4    ED53 0400   SDED BUFE       ;DELETE
01E8    22 03FE     SHLD BUFA       ;POINTER
01EB    EB          XCHG
01EC    37          STC
01ED    3F          CMC
01EE    ED52        DSBC D          ;LAENGE
01F0    E5          PUSH H
01F1    C1          POP B
01F2    2A 03FE     LHLD BUFA
01F5    E5          PUSH H
01F6    D1          POP D
01F7    23          INX H
01F8    EDB0        LDIR            ;DELETE AUSFUEHREN
01FA    2A 03FE     LHLD BUFA       ;POINTER ZURUECK
01FD    ED5B 0400   LDED BUFE
0201    1B          DCX D           ;POINTER-1
0202    C9          RET
                    ;
                    ;
                    ; * HAUPTPROGRAMM *
                    ;
0203                ENTRY:          ;NEUSTART
0203    2A 03FE     LHLD BUFA       ;AKTUELLER ZEIGER
0206    ED5B 0400   LDED BUFE       ;BUFFERENDE
020A    180B        JMPR MLOOP
020C                START:
020C    21 1000     LXI H,BUF
020F    22 03FE     SHLD BUFA       ;POINTER DEFINIEREN
0212    22 0400     SHLD BUFE
0215    E5          PUSH H
0216    D1          POP D
                    ;
0217                MLOOP:
0217    22 03FE     SHLD BUFA       ;ZEIGER AKTUALISIEREN
021A    ED53 0400   SDED BUFE
021E    0E0D        MVI C,0DH
0220    CD 0109     CALL CO
0223    0E0A        MVI C,0AH
0225    CD 0109     CALL CO
0228    AF          XRA A
0229    32 0408     STA VORZ
022C    32 0407     STA ZAHL+1
022F    3C          INR A
0230    32 0406     STA ZAHL
0233    0E2A        MVI C,'*'
0235    CD 0109     CALL CO
0238                MLOOP1:
```

zu Abb. 2.2.2-3

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 5
.MAIN. - RDK EDITOR V1.0 790604


0238    CD 0106     CALL CI
023B    4F          MOV C,A         ;EIN ZEICHEN HOLEN
023C    CD 0109     CALL CO
023F                MLOOP2:
023F    FE45        CPI 'E'         ;ENDE EDIT
0241    CA 027A     JZ EXIT1
0244    FE42        CPI 'B'         ;BEGIN POINTER
0246    CA 028A     JZ BEGIN
0249    FE5A        CPI 'Z'         ;POINTER AUF ENDE
024B    CA 0284     JZ ZENDE
024E    FE0D        CPI 0DH
0250    CA 0217     JZ MLOOP
0253    FE0A        CPI 0AH
0255    CA 0217     JZ MLOOP        ;FEHLERFALL
0258    FE49        CPI 'I'         ;INSERT
025A    CA 036C     JZ INSERT
025D    FE4B        CPI 'K'         ;KILL
025F    CA 0366     JZ KILL
0262    FE4C        CPI 'L'         ;LINE
0264    CA 03D5     JZ LINE
0267    FE54        CPI 'T'         ;TYPE                    zu Abb. 2.2.2-3
0269    CA 0308     JZ TYPE
026C    FE20        CPI ' '
026E    C2 0297     JNZ ZAHLEIN     ;EINGABE ZAHL
0271    CD 0148     CALL LINEF      ;ERST VORSCHUB
0274    CD 012B     CALL PRINTS     ;AKTUELLE ZEILE
0277    C3 0217     JMP MLOOP
                    ;
027A                EXIT1:          ;IN MONITOR
027A    22 03FE     SHLD BUFA       ;ABER POINTER AKTUAL.
027D    ED53 0400   SDED BUFE
0281    C3 0112     JMP EXIT
                    ;
0284                ZENDE:
                                    ;POINTER AUF ENDE
0284    2A 0400     LHLD BUFE       ;ENDE POINTER
0287    C3 0217     JMP MLOOP
                    ;
028A                BEGIN:
                                    ;POINTER AUF ANFANG
028A    21 1000     LXI H,BUF       ;ANFANGSPOINTER
028D    C3 0217     JMP MLOOP
                    ;
                    ;
                    ;
                    ;
0290                ECHO:
0290    CD 0106     CALL CI
0293    4F          MOV C,A
0294    C3 0109     JMP CO
                    ;
0297                ZAHLEIN:        ;EINGABE EINER# ZAHL
0297    E5          PUSH H
0298    D5          PUSH D
0299    C5          PUSH B          ;RETTEN
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 6
.MAIN. - RDK EDITOR V1.0 790604


029A    F5          PUSH PSW
029B    AF          XRA A
029C    32 0408     STA VORZ
029F    32 0407     STA ZAHL+1
02A2    3C          INR A
02A3    32 0406     STA ZAHL                                    zu Abb. 2.2.2-3
02A6    F1          POP PSW
02A7    1803        JMPR LPZA1
                    ;
02A9                LPZA:
02A9    CD 0290     CALL ECHO       ;BENUTZERZEICHEN
02AC                LPZA1:
02AC    32 0402     STA TEMP        ;ZWISCHENSPEICHERN
02AF    FE2D        CPI '-'
02B1    C2 02BB     JNZ CONZA
02B4    3EFF        MVI A,0FFH
02B6    32 0408     STA VORZ        ;NEGATIVE ZAHL
02B9    18EE        JMPR LPZA       ;NEUES ZEICHEN
02BB                CONZA:
02BB    FE2B        CPI '+'         ;IGNORE +
02BD    28EA        JRZ LPZA
02BF    FE30        CPI '0'
02C1    DA 0217     JC MLOOP        ;UNDEF ZEICHEN
02C4    FE3A        CPI 3AH
02C6    D2 0217     JNC MLOOP
02C9    21 0000     LXI H,0         ;ANFANGSWERT
02CC    44          MOV B,H
02CD                TN1:
02CD    FE30        CPI '0'
02CF    DA 02FF     JC FINZ
02D2    FE3A        CPI 3AH
02D4    D2 02FF     JNC FINZ
02D7    3EF0        MVI A,0F0H
02D9    A4          ANA H
02DA    C2 02FF     JNZ FINZ
02DD    04          INR B
02DE    C5          PUSH B
02DF    44          MOV B,H
02E0    4D          MOV C,L
02E1    29          DAD H
02E2    29          DAD H
02E3    09          DAD B
02E4    29          DAD H           ;MULT 10
02E5    3A 0402     LDA TEMP
02E8    F5          PUSH PSW
02E9    CD 0290     CALL ECHO       ;NEUES ZEICHEN
02EC    32 0402     STA TEMP
02EF    F1          POP PSW
02F0    E60F        ANI 0FH
02F2    85          ADD L
02F3    6F          MOV L,A
02F4    3E00        MVI A,0
02F6    8C          ADC H
02F7    67          MOV H,A
02F8    C1          POP B
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 7
.MAIN. - RDK EDITOR V1.0 790604


02F9    3A 0402     LDA TEMP
02FC    F2 02CD     JP TN1
02FF                FINZ:
02FF    22 0406     SHLD ZAHL       ;WERT RETTEN
0302    C1          POP B
0303    D1          POP D
0304    E1          POP H
0305    C3 023F     JMP MLOOP2
                    ;
                    ;                                   zu Abb. 2.2.2-3
                    ;
                    ; EINZELNE ROUTINEN
                    ;
0308                TYPE:           ;AUSDRUCKEN TEXT
0308    0E20        MVI C,' '       ;BLANK AUSGEBEN
030A    CD 0109     CALL CO
030D    22 03FE     SHLD BUFA       ;ZWISCHENSPEICHERN
0310    E5          PUSH H          ;RETTEN ALTEN POINTER
0311    3A 0408     LDA VORZ
0314    B7          ORA A
0315    C2 0337     JNZ TYPN        ;NEGATIV
0318    ED4B 0406   LBCD ZAHL       ;ANZAHL
031C                LPTY:
031C    C5          PUSH B
031D    CD 0115     CALL PRINT
0320    CD 0148     CALL LINEF      ;DRUCKEN UND VORSCHUB
0323    C1          POP B
0324    0B          DCX B           ;BEI 0 ALLES DRUCKEN
0325    78          MOV A,B
0326    B1          ORA C
0327    CA 0333     JZ MCONT
032A    CD 010C     CALL CSTC       ;CONSOL STATUS TESTEN
032D    B7          ORA A           ;0 WENN NICHTS DA
032E    C2 0333     JNZ MCONT       ;ABBRUCH SONST
0331    18E9        JMPR LPTY
                    ;
0333                MCONT:
0333    E1          POP H           ;ALTER POINTER
0334    C3 0217     JMP MLOOP
                    ;
                    ;
0337                TYPN:           ;RUECKWAERTS
0337    ED4B 0406   LBCD ZAHL
033B                LPTYN:
033B    C5          PUSH B
033C    CD 0158     CALL LINEM
033F    C1          POP B
0340    0B          DCX B
0341    78          MOV A,B
0342    B1          ORA C
0343    C2 033B     JNZ LPTYN       ;BIS N AUSGEFUEHRT
                                    ;IN HL IST ANFANG
0346    ED4B 03FE   LBCD BUFA       ;ENDE VERGLEICH
034A                LPTNY:
034A    7C          MOV A,H
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 8
.MAIN. - RDK EDITOR V1.0 790604


034B    B8          CMP B           ;ENDE VERGLEICH
034C    DA 0354     JC CONLTYN
034F    7D          MOV A,L
0350    B9          CMP C
0351    D2 0333     JNC MCONT       ;FERTIG
0354                CONLTYN:
0354    C5          PUSH B
0355    CD 0115     CALL PRINT      ;SONST DRUCKEN
0358    CD 0148     CALL LINEF
035B    C1          POP B
035C    CD 010C     CALL CSTC
035F    B7          ORA A
0360    C2 0333     JNZ MCONT       ;WENN TASTE DANN ABBRUCH
0363    C3 034A     JMP LPTNY
                    ;
                    ;
                    ;
0366                KILL:
                                    ;NUR EINE ZEILE
0366    CD 0176     CALL KILLEX
0369    C3 0217     JMP MLOOP
                    ;
036C                INSERT:         ;EINFUEGEN VONTEXT
036C    CD 0106     CALL CI ;ZEICHEN HOLEN
036F    FE1A        CPI 1AH         ;CTRL Z IST ENDE
0371    CA 0217     JZ MLOOP
0374    FE08        CPI 08          ;BACKSTEP
0376    CA 0390     JZ BACKST
0379    FE7F        CPI 7FH         ;RUBOUT
037B    CA 0390     JZ BACKST
037E    4F          MOV C,A         ;AUSGEBEN
037F    CD 0109     CALL CO
0382                INLPA:
0382    F5          PUSH PSW        ;RETTEN
0383    CD 01C5     CALL INP        ;EINFUEGEN
0386    F1          POP PSW
0387    77          MOV M,A         ;ZEICHEN ABLEGEN
0388    23          INX H           ;POINTER+1
0389    FE0D        CPI 0DH         ;CR
038B    CA 03CE     JZ LFEED
038E    18DC        JMPR INSERT
0390                BACKST:         ;LOESCHT EIN ZEICHEN
0390    7C          MOV A,H         ;ANFANG PRUEFEN
0391    FE10        CPI BUFH
0393    C2 039C     JNZ COBAC
0396    7D          MOV A,L
0397    FE00        CPI BUFL        ;
0399    CA 036C     JZ INSERT       ;KEIN DELETE AUSFUEHREN
039C                COBAC:
039C    2B          DCX H           ;EIN ZEICHEN ZURUECK
039D    7E          MOV A,M
039E    FE0A        CPI 0AH
03A0    CA 03BE     JZ LFIN         ;LINEFEED
03A3    F5          PUSH PSW
03A4    CD 01E4     CALL INM        ;DELETE
```

zu Abb. 2.2.2-3

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                        PAGE 9
.MAIN. - RDK EDITOR V1.0 790604


03A7    F1          POP PSW         ;ZEICHEN NOCH DA
03A8    FE0D        CPI 0DH         ;CR
03AA    CA 03CC     JZ CRIN
03AD    0E08        MVI C,8
03AF    CD 0109     CALL CO         ;LOESCHEN ZEICHEN
03B2    0E20        MVI C,20H
03B4    CD 0109     CALL CO
03B7    0E08        MVI C,8
03B9    CD 0109     CALL CO
03BC    18AE        JMPR INSERT
03BE                LFIN:           ;LINE FEED AUFGETRETEN
03BE    E5          PUSH H
03BF    23          INX H           ;NACH LF POSITIONIEREN
03C0    CD 0158     CALL LINEM
03C3    CD 012B     CALL PRINTS     ;ERST AUSGEBEN
03C6    E1          POP H
03C7    CD 01E4     CALL INM        ;LOESCHEN
03CA    18C4        JMPR BACKST     ;CR LOESCHEN
                    ;
03CC                CRIN:
03CC    189E        JMPR INSERT     ;WEITER EINFUEGEN
                    ;
                    ;
                    ;
03CE                LFEED:
03CE    0E0A        MVI C,0AH
03D0    CD 0109     CALL CO
03D3    18AD        JMPR INLPA
                    ;
03D5                LINE:
03D5    3A 0408     LDA VORZ
03D8    B7          ORA A           ;ZEILENVORSCHUB
03D9    C2 03ED     JNZ LINEG       ;NEGATIVE
03DC    ED4B 0406   LBCD ZAHL       ;ANZAHL
03E0                LPLIN:
03E0    C5          PUSH B
03E1    CD 0148     CALL LINEF
03E4    C1          POP B
03E5    0B          DCX B
03E6    78          MOV A,B
03E7    B1          ORA C
03E8    CA 0217     JZ MLOOP
03EB    18F3        JMPR LPLIN
                    ;                       zu Abb. 2.2.2-3
03ED                LINEG:
03ED    ED4B 0406   LBCD ZAHL
03F1                LPIN:
03F1    C5          PUSH B
03F2    CD 0158     CALL LINEM
03F5    C1          POP B
03F6    0B          DCX B
03F7    78          MOV A,B
03F8    B1          ORA C
03F9    CA 0217     JZ MLOOP
03FC    18F3        JMPR LPIN
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 10
.MAIN. - RDK EDITOR V1.0 790604


                        ;
03FE    0000            BUFA: .WORD 0
0400    0000            BUFE: .WORD 0
0402    0000            TEMP: .WORD 0
0404    0000            TEMP1:.WORD 0
0406    0000            ZAHL: .WORD 0
0408    00              VORZ: .BYTE 0                    zu Abb. 2.2.2-3
1000                    BUF=1000H
0010                    BUFH=10H
0000                    BUFL=0
                        ;
                        .END
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 11
.MAIN. - RDK EDITOR V1.0 790604
+++++ SYMBOL TABLE +++++

BACKST 0390        BEGIN  028A        BUF    1000        BUFA   03FE
BUFE   0400        BUFH   0010        BUFL   0000        CI     0106
CO     0109        COBAC  039C        COLPM  016D        CONLTY 0354
CONFR2 0136        CONTP  0120        CONZA  02BB        CRIN   03CC
CSTC   010C        ECHO   0290        ENTRY  0203        EXIT   0112
EXIT1  027A        FINFR  0129        FINZ   02FF        INLPA  0382
INM    01E4        INP    01C5        INSERT 036C        KCO    0183
KI1    0179        KILL   0366        KILLEX 0176        KK1    018A
LCON   0150        LFEED  03CE        LFIN   03BE        LINE   03D5
LINEG  03ED        LINEM  0158        LINEP  0148        LMCON  0162
LP     010F        LPIN   03F1        LPLIN  03E0        LPMC   0163
LPTNY  034A        LPTY   031C        LPTYN  033B        LPZA   02A9
LPZA1  02AC        MCONT  0333        MLOOP  0217        MLOOP1 0238
MLOOP2 023F        PR1    0116        PR2    012C        PRINT  0115
PRINTS 012B        START  020C        TEMP   0402        TEMP1  0404
TN1    02CD        TYPE   0308        TYPN   0337        VORZ   0408
ZAHL   0406        ZAHLEI 0297        ZENDE  0284        .BLNK. 0000:03 X
.DATA. 0000*    X  .PROG. 0000'    X
```

schon bekannte Sprungtabelle der Vektoren zu den Ein- und Ausgabe-Routinen der Konsole. Als erster Eintrag steht der Sprung zur Stelle START, der mindestens einmal angesprochen werden muß. Der zweite Eintrag ist ein Sprung zur Stelle ENTRY, die einen erneuten Start des EDITORS bewirkt, wenn dieser z.B. mit E oder RESET verlassen wurde, und der alte Text noch weiter editiert werden soll. CI, CO, CSTS entsprechen den Routinen aus dem vorhergehenden Abschnitt. LP ist ein Sprung zu einem Drucker, der aber auch auf CO zeigen kann. Er wurde hier noch nicht weiter verwendet. EXIT schließlich ist der Sprung zum neuen Start des Monitors.

Unterprogramme

PRINT bewirkt den Ausdruck einer Zeile bis zum Zeichen LF, falls die Zeile nicht über die Stelle BUFE hinausreicht. BUFA ist im Registerpaar HL enthalten und zeigt auf die aktuelle Zeile. Im Register DE steht BUFE und wird zur Ende-Überwachung eingesetzt. PRINTS wirkt wie PRINT, nur daß das am Schluß stehende CR LF nicht mit ausgegeben wird. Die Routine wird später z.B. für das Teilprogramm INSERT verwendet.

LINEP schiebt den Pointer BUFA um eine Zeile in positiver (Textende) Richtung. Dabei wird entweder bis zum nächsten LF erhöht oder bis zum Bufferende BUFE.

LINEM erniedrigt BUFA um soviel Stellen, daß entweder nur eine Zeile zurück oder bis zum Bufferanfang gegangen wird.

KILLEX löscht eine Zeile. Diese Zeile wird durch BUFA bestimmt. Das Ende der Zeile ist entweder durch LF oder durch BUFE bestimmt. Der gesamte Textspeicher wird dazu um die entsprechende Anzahl von Bytes verschoben, wobei ein Befehl des Z80-Blockmove zu Hilfe genommen wird. BUFE wird anschließend aktualisiert.

INP fügt an die durch BUFA angegebene Stelle ein Zeichen in den Textbuffer ein.
INM löscht ein Zeichen aus dem Textspeicher an der Stelle BUFA.

Hauptprogramm

HL enthält normalerweise den Zeiger BUFA und DE den Zeiger BUFE. Beim Starten des EDITORs über ENTRY werden diese beiden Zeiger aus den gleichnamigen Speicherzellen geladen, so daß der Editiervorgang ohne Datenverlust weitergeführt werden kann. Bei START werden BUFA und BUFE mit BUF belegt. Das kommt einem Löschen des Textspeichers gleich.

MLOOP ist die Hauptschleife des EDITORs. Nach Eintritt werden die Speicherzellen BUFA und BUFE aktualisiert, so daß z.B. nach RESET die beiden Pointer definiert sind, so daß ein erneuter Start über ENTRY erfolgen kann. Der Zahlenspeicher wird mit 1 belegt, so daß z.B. T allein genügt, um eine einzige Zeile auszugeben. Das Vorzeichen wird mit 0 für + belegt. Bei MLOOP1 wird ein Zeichen vom Benutzer eingelesen. Anschließend wird mit einer Vergleichskette das Zeichen einem Befehl zugeordnet. Die Routine ZAHLEIN bewirkt die Eingabe einer gegebenenfalls mit einem Vorzeichen versehenen Zahl. Das Ergebnis wird in den Zellen ZAHL und VORZ abgelegt. Bei korrekter Eingabe wird nach Erkennung eines Buchstabens an die Stelle MLOOP2 gesprungen. Von dort wird mit Hilfe der Vergleichskette der Befehl ausgeführt.

Teilprogramme

Das Programm TYPE hat zwei Möglichkeiten zu unterscheiden. Das Vorzeichen kann negativ oder positiv sein. Ist das Vorzeichen negativ, so wird zum Programmteil TYPN gesprungen. Ist die Zahl positiv, so werden n weitere Zeilen in Richtung Bufferende ausgegeben. TYPN gibt n Zeilen in Richtung Bufferanfang aus. Dabei muß bei TYPN erst der Anfang des auszugebenden Bereichs festgestellt werden, da sonst die Zeilen in der falschen Reihenfolge ausgegeben werden würden.

INSERT ist etwas komplexer, da zahlreiche verschiedene Fälle unterschieden werden müssen. Es wird zunächst ein Zeichen von der Konsole geholt. Handelt es sich um ein

Zeichen CTRL Z (1 AH), so wird wieder in das Hauptprogramm gesprungen, denn dieses Zeichen beendet den Insert-Vorgang. Bei Backstep (Ø8H) wird in die Routine BACKSTEP gesprungen, die ein Zeichen aus dem Buffer entfernt. Das gleiche geschieht bei dem Zeichen rubout (7FH). In jedem anderen Fall wird das Zeichen an die Konsole ausgegeben und in den Buffer eingefügt.

LINE ist das Programm für den Befehl L. Auch hier muß zwischen einem negativen und einem positiven Vorzeichen unterschieden werden. Je nachdem wird LINEG angesprungen oder im Programm LINE fortgefahren.

Erweiterungen des EDITORs

Der hier gezeigte EDITOR ist eine kleine Version von größeren, handelsüblichen EDITORen. Hier fehlt z.B. die Möglichkeit nach Zeichenketten zu suchen und diese

```
!.MAIN.7F
\03.PROG.010000.DATA.020000.BLNK.0300006A
:18010000C30C02C30302C303F0C309F0C312F0C30FF0C31EF0E57CBA69
:18011800C220017DEBCA29014E23CD0901FE0A20EDE1C9E57CBAC236A6
:180130000017DEBCA29017EFE0ACA2901FE0DCA2901234FCD090118E4CC
:180148007CBAC250017DEBC87E23FE0AC8C348017CFE10C262017DFEAF
:18016000000C82B7CFE10C26D017DFE00C82B7EFE0AC26301230922FEB4
:18017800037CBAC283017DEBCA8A017E23FE0AC27901220204ED530016
:18019000042A0004ED4B0204373FED4223E5C12A0204ED5BFE03EDE063
:1801A8002A0204ED4BFE03373FED42E5D12A0004373FED52220004E58D
:1801C000D12AFE03C913ED53000422FE03EB373FED52E5C12A0004E58F
:1801D800D12BEDEB82AFE03ED5B0004C9ED53000422FE03EB373FED5227
:1801F000E5C12AFE03E5D123EDB02AFE03ED5B00041BC92AFE03ED5BE2
:18020800000418 0E21001022FE03220004E5D122FE03ED5300040E0D05
:18022000CD09010E0ACD0901AF32080432070 43C3206040E2ACD09014F
:18023800CD06014FCD0901FE45CA7A02FE42CA8A02FE5ACA8402FE0DE2
:18025000CA1702FE0ACA1702FE49CA6C03FE4BCA6603FE4CCAD503FEE2
:18026800054CA0803FE20C29702CD4801CD2B01C3170222FE03ED53008E
:18028000004C312012A0004C3170221001 0C31702CD06014FC30901E5A0
:18029800D5C5F5AF32080432070 43C320604F11803CD9002320204FE7C
:1802B0002DC2BB023EFF32080418EEFE2B28EAFE30DA1702FE3AD2178C
:1802C8000221000044FE30DAFF02FE3AD2FF023EF0A4C2FF0204C54401
:1802E0004D29290929 3A0204F5CD9002320204F1E60F856F3E008C675E
:1802F800C13A0204F2CD02220604C1D1E1C33F020E20CD090122FE0361
:18031000E53A0804B7C23703ED4B0604C5CD1501CD4801C10B78B1CA38
:18032800303CD0C01B7C2330318E9E1C31702ED4B0604C5CD5801C152
:18034000B78B1C23B03ED4BFE037CB8DA54037DB9D23303C5CD1501ED
:18035800CD4801C1CD0C01B7C23303C34A03CD7601C31702CD0601FE2B
:18037000 1ACA1702FE08CA9003FE7FCA90034FCD0901F5CDC501F17725
:1803880023FE0DCACE0318DC7CFE10C29C037DFE00CA6C032B7EFE0A50
:1803A000CABE03F5CDE401F1FE0DCACC030E08CD0901 0E20CD09010E7E
:1803B8000 8CD090118AEE523CD5801CD2B01E1CDE40118C4189E0E0A24
:1803D000CD090118AD3A0804B7C2ED03ED4B0604C5CD4801C10B78B188
:1803E800CA170218F3ED4B0604C5CD5801C10B78B1CA170218F30000FF
:090400000000000000000000F3
:0000000000
A>3FF3100 3FF
```

Abb. 2.2.2-4 Object Code des EDITORs im Intel-HEX-Format

gegebenenfalls automatisch durch andere Zeichenketten zu ersetzen. Auch fehlt die Möglichkeit, Text direkt von einem Peripherie-Gerät zu übernehmen oder dorthin auszugeben. Hier kann nur über ein Monitorprogramm ein Text von einem Peripherie-Gerät übernommen werden. Es ist dann erforderlich, die Speicherzellen BUFA und BUFE manuell auf den Text anzupassen, um dann den EDITOR über den ENTRY-Vektor zu starten.

Eine kleine Erweiterung kann leicht selbst erprobt werden. Das Zeichen tab (9), welches normalerweise bewirkt, daß in die nächste vom Anfang gezählte 8. Spalte

```
0100 C3 0C 02 C3 03 02 C3 03 F0 C3 09 F0 C3 12 F0 C3 ................
0110 0F F0 C3 1E F0 E5 7C BA C2 20 01 7D BB CA 29 01 ......\.. .]..).
0120 4E 23 CD 09 01 FE 0A 20 ED E1 C9 E5 7C BA C2 36 N#..... ....\..6
0130 01 7D BB CA 29 01 7E FE 0A CA 29 01 FE 0D CA 29 .]..).~...)....)
0140 01 23 4F CD 09 01 18 E4 7C BA C2 50 01 7D BB C8 .#O.....\..P.]..
0150 7E 23 FE 0A C8 C3 48 01 7C FE 10 C2 62 01 7D FE ~#....H.\...b.].
0160 00 C8 2B 7C FE 10 C2 6D 01 7D FE 00 C8 2B 7E FE ..+\...m.]...+~.
0170 0A C2 63 01 23 C9 22 FE 03 7C BA C2 83 01 7D BB ..c.#."..\....].
0180 CA BA 01 7E 23 FE 0A C2 79 01 22 02 04 ED 53 00 ...~#...y."...S.
0190 04 2A 00 04 ED 4B 02 04 37 3F ED 42 23 E5 C1 2A .*...K..7?.B#..*
01A0 02 04 ED 5B FE 03 ED B0 2A 02 04 ED 4B FE 03 37 ...[....*...K..7
01B0 3F ED 42 E5 D1 2A 00 04 37 3F ED 52 22 00 04 E5 ?.B..*..7?.R"...
01C0 D1 2A FE 03 C9 13 ED 53 00 04 22 FE 03 EB 37 3F .*.....S.."...7?
01D0 ED 52 E5 C1 2A 00 04 E5 D1 2A ED B8 2A FE 03 ED .R..*....+..*...
01E0 5B 00 04 C9 ED 53 00 04 22 FE 03 EB 37 3F ED 52 [....S.."...7?.R
01F0 E5 C1 2A FE 03 E5 D1 23 ED B0 2A FE 03 ED 5B 00 ..*....#..*...[.
0200 04 1B C9 2A FE 03 ED 5B 00 04 18 0B 21 00 10 22 ...*...[....!.."
0210 FE 03 22 00 04 E5 D1 22 FE 03 ED 53 00 04 0E 0D ..".....".....S....
0220 CD 09 01 0E 0A CD 09 01 AF 32 08 04 32 07 04 3C .........2..2..<
0230 32 06 04 0E 2A CD 09 01 CD 06 01 4F CD 09 01 FE 2...*.......O....
0240 45 CA 7A 02 FE 42 CA 8A 02 FE 5A CA 84 02 FE 0D E.z..B....Z.....
0250 CA 17 02 FE 0A CA 17 02 FE 49 CA 6C 03 FE 4B CA .........I.l..K.
0260 66 03 FE 4C CA D5 03 FE 54 CA 08 03 FE 20 C2 97 f..L....T.... ..
0270 02 CD 48 01 CD 2B 01 C3 17 02 22 FE 03 ED 53 00 ..H..+....."...S.
0280 04 C3 12 01 2A 00 04 C3 17 02 21 00 10 C3 17 02 ....*.....!.....
0290 CD 06 01 4F C3 09 01 E5 D5 C5 F5 AF 32 08 04 32 ...O........2..2
02A0 07 04 3C 32 06 04 F1 18 03 CD 90 02 32 02 04 FE ..<2........2...
02B0 2D C2 BB 02 3E FF 32 08 04 18 EE FE 2B 28 EA FE -...>.2.....+(..
02C0 30 DA 17 02 FE 3A D2 17 02 21 00 00 44 FE 30 DA 0....:...!..D.0.
02D0 FF 02 FE 3A D2 FF 02 3E F0 A4 C2 FF 02 04 C5 44 ...:...>.......D
02E0 4D 29 29 09 29 3A 02 04 F5 CD 90 02 32 02 04 F1 M)).):......2...
02F0 E6 0F 85 6F 3E 00 8C 67 C1 3A 02 04 F2 CD 02 22 ...o>..g.:......"
0300 06 04 C1 D1 E1 C3 3F 02 0E 20 CD 09 01 22 FE 03 ......?.. ...".. 
0310 E5 3A 08 04 B7 C2 37 03 ED 4B 06 04 C5 CD 15 01 .:....7..K......
0320 CD 48 01 C1 08 78 B1 CA 33 03 CD 0C 01 B7 C2 33 .H...x..3......3
0330 03 18 E9 E1 C3 17 02 ED 4B 06 04 C5 CD 58 01 C1 ........K....X..
0340 08 78 B1 C2 3B 03 ED 4B FE 03 7C B8 DA 54 03 7D .x..;..K..\..T.]
0350 B9 D2 33 03 C5 CD 15 01 CD 48 01 C1 CD 0C 01 B7 ..3......H......
0360 C2 33 03 C3 4A 03 CD 76 01 C3 17 02 CD 06 01 FE .3..J..v........
0370 1A CA 17 02 FE 08 CA 90 03 FE 7F CA 90 03 4F CD ..............O.
0380 09 01 F5 CD C5 01 F1 77 23 FE 0D CA CE 03 18 DC .......w#.......
0390 7C FE 10 C2 9C 03 7D FE 00 CA 6C 03 2B 7E FE 0A \.....]...l.+~..
03A0 CA BE 03 F5 CD E4 01 F1 FE 0D CA CC 03 0E 08 CD ................
03B0 09 01 0E 20 CD 09 01 0E 08 CD 09 01 18 AE E5 23 ... ...........#
03C0 CD 58 01 CD 2B 01 E1 CD E4 01 18 C4 18 9E 0E 0A .X..+...........
03D0 CD 09 01 18 AD 3A 08 04 B7 C2 ED 03 ED 4B 06 04 .....:.......K..
03E0 C5 CD 48 01 C1 08 78 B1 CA 17 02 18 F3 ED 4B 06 ..H...x.......K.
03F0 04 C5 CD 58 01 C1 08 78 B1 CA 17 02 18 F3 00 00 ...X...x........
```

Abb. 2.2.2-5 EDITOR Ausdruck für Adresse 100H

gesprungen wird, ist hier nicht wirksam. Es wird zwar vom EDITOR richtig abgespeichert, aber das Zeichen wird auch auf die Konsole weitergeleitet. Es ist nun eine Routine zu schreiben, die im Befehl INSERT die Anzahl der eingegebenen, sichtbaren Zeichen überwacht und Tabulatoren entsprechend in Leerzeichen (blank) umsetzt. Es ist außerdem eine Modifizierung der Funktion TYPE vorzunehmen, da beim Ausgeben des Textes auch Tabulatoren expandiert werden sollen. Eine weitere Schwierigkeit ergibt sich bei der Funktion BACKSTEP. Sind nämlich Tabulatoren im Text, so muß der Cursor um die entsprechende Anzahl von Zeichen blank nach links zurückrücken. Dies kann z.B. mit einem kleinen Zeilenpuffer erreicht werden, der eine Zeile speichert. Wenn jetzt zurückgeschritten wird, so liegt die Anzahl der Leerzeichen in diesem Puffer, der nur die blank-Zeichen enthält.

Abb. 2.2.2-4 zeigt den Objektcode des EDITORs im INTEL-HEX-Format und *Abb. 2.2.2-5* zeigt den HEX-Ausdruck des EDITORs.

2.2.3 Verschiedene EDITOR-Ausführungen

Allgemein kann zwischen Zeilen- und Block-orientierten EDITORen unterschieden werden. Der im vorhergehenden Abschnitt behandelte EDITOR war zeilenorientiert. Kennzeichnend für diesen ist, daß immer nur eine Zeile verändert werden kann. Im Gegensatz ist bei dem Block-orientierten EDITOR z.B. eine ganze Bildschirmseite sichtbar. Mit Hilfe des Cursors kann beliebig in diesem Feld positioniert werden. Es können einfache Control-Zeichen verwendet werden, um neue Zeichen einzufügen oder alte zu löschen. Dabei wird nicht ein Meldezeichen vom EDITOR auf den Bildschirm ausgegeben. Meistens kann ferner „geblättert" werden. Dabei wird durch ein Steuerzeichen erreicht, daß die nächste oder vorhergehende Textseite auf dem Bildschirm erscheint.

2.3 Assembler

Der Assembler ist das wichtigste Werkzeug bei der Programmentwicklung. Er hat die Aufgabe, ein in mnemonischen Bezeichnungen (leicht zu behaltende Bezeichnungen) geschriebenes Programm in die Maschinensprache zu übersetzen. Dies erfolgt meist in zwei Durchläufen. In einem ersten Durchlauf, in dem das gesamte Programm vom Assembler eingelesen wird, werden nur alle Marken betrachtet. Diese werden mit der schon jetzt bekannten Programmadresse in eine Tabelle eingetragen, die absolut oder relativ sein kann. Mehrfach vorkommende Marken werden als Fehler erkannt und führen zu einer Fehlermeldung auf der Konsole. In einem zweiten Durchlauf werden die einzelnen mnemonischen Angaben in den Maschinencode übersetzt. Ebenso werden symbolische Namen übersetzt. Dazu wird in der Symboltabelle nach dem Namen gesucht und entsprechend der Zuordnungstabelle (1 oder 2 Byte) in den Objektcode eingefügt, falls er gefunden wird. In der Symboltabelle werden auch Konstanten abgelegt, die mit EQU oder = definiert wurden. Die Übersetzung der mnemonischen Ausdrücke in den Objektcode erfolgt auch über eine Tabelle. Diese wird dann ebenfalls durchsucht und der Objektcode entnommen, falls ein Begriff hier gefunden wird. Weitere Informationen in dieser Tabelle geben Aufschluß über die Länge des Maschinenbefehls, Adressierart und Masken.

2.3.1 Relocating Assembler

Bei einem Relocating Assembler muß in dem geschriebenen Programm noch nicht festliegen, auf welche Adresse das Programm gelegt werden soll. Der Assembler erzeugt dazu eine spezielle Objektcode-File, die dann – nach der Assemblierung mit einem weiteren Programm – dem LADER – auf eine bestimmte Adresse gelegt werden kann. In der Objektcode-File ist neben dem Maschinencode auch noch Information abgelegt, an welchen Stellen des Programms Adressen stehen, bezogen auf eine relative Anfangsadresse Ø. Der LADER wertet dann diese Information aus, um an die betreffenden Stellen nach Angabe einer absoluten Anfangsadresse eine Konstante zu den Adressen im Programm zu addieren, die z.B. bei Sprungbefehlen vorkommen. Das schon verwendete TDL Relocating Format ist eine solche relocalisierbare File.
Das Format ist folgendermaßen definiert:

Zeichen	Erklärung
0 . . . 1	CR LF, um Blöcke voneinander zu trennen.
2	Strichpunkt kennzeichnet den Anfang eines relocalisierbaren Blocks.
3 . . . 4	Dieses Byte wird durch zwei 7-Bit-Zeichen nach DIN 66003 (ASCII) 0 . . . 9, A . . . F gibt die Anzahl der Daten-Bytes an. Z.B. sedezimal 19 ($19_{(16)}$) an dieser Stelle bedeutet, es sind dezimal gerechnet in diesem Block 25 ($25_{(10)}$) Daten-Bytes.
5 . . . 8	In den Bytes ist die relative Anfangsadresse des Blocks untergebracht. Sie wird beginnen mit dem höherwertigen Byte angegeben.
9 . . . 10	Dieses Byte enthält die Relocalisierungsinformation. Ist dieses Byte 0, so wird die angegebene Anfangsadresse absolut verwendet. Ist das Byte 1, so wird die Adresse durch den Relocalisierungsfaktor modifiziert.
11 . . . 12	Dieses Byte enthält die Relocalisierungsinformation für die nächsten 8 Byte der Daten. Dabei entspricht jedes Bit in diesem Byte einem Byte der nächsten 8 Byte der Daten. Bit 7 ist für das erste Byte zuständig, Bit 0 für das letzte. Ist das betreffende Bit 0, so bedeutet das, das Byte wird unverändert gelassen. Ist das Bit 1, gefolgt von einem Bit mit 0, so sind die beiden dazugehörigen Bytes eine relocalisierbare Adresse, und der Relocalisierungsfaktor muß hinzuaddiert werden. Ein auf 1 gesetztes Bit, welches von einem ebenfalls auf 1 gesetzten Bit gefolgt wird, stellt die Information für den LINKING LOADER dar, der noch besprochen wird.
13 . . . 28	Hier sind die Bytes für die Daten enthalten. Die Gesamtzahl dieser Bytes ist durch die Information am Anfang des Blocks festgelegt. Nach jedem achten Byte der Daten wird aber wieder die Relocalisierungsinformation abgelegt. Das erfolgt dann, wenn es sich um einen Relocating Block handelt und das Zeichen 9, 10 nicht 0 sondern 1 ist.

N . . . (N+1) Dieses Byte enthält die Blockprüfsumme, die dem Zweierkomplement der Summe über alle vorhergehenden Bytes in diesem Block entspricht, so daß die Summe über alle Bytes immer 0 ergeben muß.

Abb. 2.3.1-1 zeigt ein Beispiel eines im Relocating Format geschriebenen Programms. Mit der Anwendung .PREL wird dem Assembler mitgeteilt, daß er die Object File in diesem Relocating Format erzeugen soll.

.PHEX gibt an, daß die Object File in einem direkt lesbaren Format geschrieben wird. Im Gegensatz dazu wäre bei .PBIN die Object File dual abgelegt worden, ohne daß die einzelnen Bytes in zwei Sedezimalziffern aufgeteilt worden wären. Als Ende Zeichen gilt bei dem HEX-Format die Folge: ØØØØØØØØ. Die im Bild sichtbaren Folgen, die mit ! beginnen, sind Informationen für den Linking Loader, die hier überflüssig sind.

Abb. 2.3.1-1 Relocalisierbares Programm

```
                                .PREL
                                .PHEX
                                ; RELATIVE FILE
0000'                           ANF:
0000'   77                      MOV M,A
0001'                           LOOP:
0001'   C3 0001'                JMP LOOP
                                .END
```

```
!.MAIN.7F
!03.PROG.010004.DATA.020000.BLNK.03000066
:0500000120772C301009F
:0000000000
A>
```

2.3.2 Linking Assembler

Bei einem Linking Assembler kann ein Programm in mehrere Teilprogramme zerlegt werden. Dabei werden die Teilprogramme getrennt übersetzt. Sie können auch Bezüge auf Marken und Speicherzellen enthalten, die nicht im gleichen Programmteil definiert sind. Der Vorteil dieser Programmtechnik ist, daß z.B. bei großen Programmen bei Änderungen nicht das gesamte Programm übersetzt werden muß, sondern nur das Teilprogramm mit den Änderungen. Am Schluß müssen die Teilprogramme mit einem Linking Loader zu einem lauffähigen Programm gebunden werden. Dieser legt nicht nur das Programm auf eine bestimmte Adresse, sondern er setzt auch die richtigen Adressen in die Teilprogramme ein. Sie waren vorher nicht bekannt, weil sie in einem anderen Programm festgelegt wurden.

Dazu muß der Linking Loader nicht nur die Stellen kennen, an denen eine Adresse eingesetzt werden soll, sondern auch die Namen, mit denen sie im Assemblerprogramm aufgerufen werden. Die Definition der Namen muß ebenfalls bekannt sein, so daß der Name mit dem wahren Adreßwert zur Verfügung stehen muß.

In einem Assemblerprogramm würde eine nicht definierte Marke normalerweise als Fehler erkannt werden. Um dies zu vermeiden, muß dem Assembler mit einer speziellen Anweisung bekanntgegeben werden, daß die betreffende Marke in einem anderen Programm definiert ist. Dies geschieht bei dem TDL Assembler z.B mit dem Befehl

```
                        .PREL
                        .PHEX
                        .INTERN START
                        .EXTERN SPEICHER
0000'                   ANF:
0000'    01 0000:04     LXI B,SPEICHER
0003'                   START:
0003'    02             STAX B
0004'    C9             RET
                        .END


TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                        PAGE 2
.MAIN. -
+++++ SYMBOL TABLE +++++

ANF     0000'          SPEICH 0000:04 X     START   0003'    I   .BLNK. 0000:03 X
.DATA. 0000*    X      .PROG. 0005'    X

*I.MAIN.7F
\04.PROG.010005.DATA.020000.BLNK.030000SPEICH040000A4
#01START 0100034D
:0700000160010400000C9C8
:0000000000
A>
```

Abb. 2.3.2-1 Programm mit INTERN und EXTERN Bezügen

.EXTERN. In dem Programm, in welchem eine Marke definiert wird und von der bekannt ist, daß sie in einem anderen Programm verwendet wird, muß umgekehrt die Anweisung .INTERN gegeben werden. Diese Anweisung veranlaßt, daß eine entsprechende Information in der Object File abgelegt wird, so daß der Linking Loader später die Marke erkennt. *Abb. 2.3.2-1* zeigt ein Beispiel eines Programms, daß die Anweisungen .INTERN und .EXTERN verwendet. Hierbei ist die Marke START mit der Anweisung .INTERN einem anderen Programm zugänglich gemacht und der Name SPEICHER wird mit der Anweisung .EXTERN als nicht im Teilprogramm vorkommend definiert. In der Objektcode File werden mit alle externen Bezüge angegeben und mit # alle internen Definitionen. Den externen Bezügen werden aufsteigende Nummern verliehen, die es später dem Linking Loader ermöglichen, die Namen an der richtigen Stelle durch die richtigen Adressen zu ersetzen. .PROG. sowie .DATA. und .BLNK. werden vom Assembler automatisch als extern definiert und dienen der Einteilung in ein Block-Programm und in einen davon getrennten Block DATA. Daher ist es möglich, Daten- und Programmbereich auf getrennten Adressen unterzubringen.

2.3.3 Makro Assembler

Kommt ein Programmabschnitt mehrmals in einem Programm vor, so wird im allgemeinen ein Unterprogramm verwendet. Die Makro-Programmiertechnik gleicht fast dieser Unterprogrammtechnik, da sie auch nur dann angewendet wird, wenn ein Programmabschnitt mehrmals verwendet wird. Nur hierbei wird im Assembler der Programmabschnitt unter einem Namen bekannt gemacht. Der Name kann dann anstelle der Programmsequenz

```
                        .PREL
                        .PHEX
                        ; MAKRODEFINITON

                        .DEFINE MOVE[A,E,Z]=
                        [
                        .IFG E-A, [
                        LXI D,Z             ;ZIEL
                        LXI H,A             ;ANFANG
                        LXI B,E-A           ;ENDE-ANFANG IST LAENGE
                        LDIR                ;INCREMENT BLOCK MOVE
                        ]
                        .IFL E-A, [
                        LXI D,Z
                        LXI H,A
                        LXI B,A-E           ;ANFANG-ENDE IST POSITIV
                        LDDR                ;DECREMENT BLOCK MOVE
                        ]
                        .IFE E-A, [
                        .ERROR /ENDE=ANFANG IN PARAMETER/
                        ]

                        ]
                        ; ENDE MAKRODEFINITION
                        ;
0000'                   START:
                        MOVE 10,20,30       ;MAKROAUFRUF
0000'   11 001E        +LXI D,30                           ;ZIEL
0003'   21 000A        +LXI H,10                   ;ANFANG
0006'   01 000A        +LXI B,20-10        ;ENDE-ANFANG IST LAENGE
0009'   EDB0           +LDIR               ;INCREMENT BLOCK MOVE
                        +]
                         MOVE 15,2,3000[
000B'   11 0BB8        +LXI D,3000
000E'   21 000F        +LXI H,15
0011'   01 000D        +LXI B,15-2         ;ANFANG-ENDE IST POSITIV
0014'   EDB8           +LDDR               ;DECREMENT BLOCK MOVE
                        +]
                         MOVE 200,200,300[
*                       +.ERROR ?/ENDE=ANFANG IN PARAMETER/
                        +]
                         ;
                         .END
```

Abb. 2.3.3-1 Programm mit Makro-Definitionen

wie ein normaler Maschinenbefehl verwendet werden. Bei jedem Aufruf ersetzt der Assembler diesen Namen durch die vorher definierte Assemblersequenz für den Programmabschnitt. Dieser Vorgang wird auch mit Makroexpansion bezeichnet. Nun wäre das aber nur von geringem Vorteil, denn diese Art Unterprogramme zu umgehen, erfordert viel Speicherplatz. Sie ist auch nur für bestimmte Anwendungsfälle (zeitkritische Vorgänge) brauchbar. Aber hier gibt es gegenüber den Unterprogrammen noch einen großen Unterschied. Der Makrodefinition können Parameter beigefügt werden, und es können Fallunterscheidungen, z.B. in Abhängigkeit dieser Parameter, eingefügt werden. Beim Aufruf des Makros durch seinen Namen mit Parametern werden je nach Parameter unterschiedliche Assemblersequenzen erzeugt.

Abb. 2.3.3-1 zeigt das Beispiel eines Programms, welches Makros verwendet. Dabei wird ein neuer Befehl mit dem Namen MOVE definiert. Er soll die Aufgabe haben, einen

Block von Daten von einer Anfangsadresse zu einer Endadresse nach einer Zieladresse zu verschieben. Dabei sollen zwei Fälle unterschieden werden. Einmal soll Endadr größer sein als Anfadr. Dann soll ein Befehl LDIR verwendet werden, wenn das nicht der Fall ist, also, wenn Anfadr größer als Endadr ist, dann LDDR. Sind aber beide Adressen gleich, so soll Fehlermeldung ausgegeben werden.

2.4 BASIC

Das Programmieren in Assembler-Sprache ist oft sehr mühsam. Deshalb wurden höhere Programmiersprachen geschaffen. Eine der einfachsten höheren Programmiersprachen stellt BASIC (Beginners all purpose symbolic instruction code) dar. Mit einem Befehl einer höheren Programmiersprache können komplexe Zusammenhänge beschrieben werden. Es ist dann nicht nötig, sich um einzelne Bits und Bytes zu kümmern. Dabei werden die Programme um einiges übersichtlicher.

2.4.1 RDK BASIC

Es soll im folgenden ein vom Verfasser abgeleitetes vereinfachtes BASIC beschrieben werden, das eine Teilmenge des „echten" BASIC darstellt. So wird auch einem Anwender mit einem Computer und wenig Speicherplatz ermöglicht, Programmerfahrung mit BASIC zu gewinnen. BASIC dieser Art benötigt etwa 3 KByte Speicherplatz sowie mindestens 1 KByte für das Anwenderprogramm. Es handelt sich um ein sogenanntes TINY BASIC, d.h. mit eingeschränkten Arithmetik-Operationen. Hier kann z.B. nur mit Festkomma und einem Bereich von $\pm$ 32767 gerechnet werden. Auch ist die String-Verarbeitung (Zeichenketten-Verarbeitung) gegenüber der universellen Sprache BASIC vereinfacht.

Grundsätzlich gibt es bei der Realisierung einer höheren Programmiersprache zwei verschiedene Möglichkeiten. Eine davon ist die Anwendung von einem COMPILER. Dieser übersetzt ein in einer höheren Programmiersprache geschriebenes Programm in Maschinen-Code, der dann anschließend gestartet wird. Die andere ist der INTERPRETER. Dieser übersetzt das Anwenderprogramm nicht in die Maschinensprache, sondern führt jeden Befehl der höheren Programmiersprache direkt aus, sobald er ihn erkannt hat. Es können zwei verschiedene Arten INTERPRETER unterschieden werden. Bei der einfachen Art wird der Befehl Buchstabe für Buchstabe analysiert und mit einer Tabelle verglichen, jedesmal, wenn ein Befehl der höheren Programmiersprache durchlaufen wird.

Bei der anderen Art wird zunächst das gesamte Programm auf diese sogenannten reservierten Wörter untersucht. Jeder Befehl wird dann in einen Zwischencode übersetzt. Beim Starten des Programms braucht dann nur dieser Zwischencode analysiert zu werden, der z.B. aus einem Byte besteht, mit dem höherwertigen Bit auf 1 gesetzt. Der Vorteil liegt in einem geringeren Speicherbedarf für das Anwenderprogramm und in einer höheren Ausführungsgeschwindigkeit.

Der im folgenden beschriebene BASIC-Interpreter ist von der ersten Art, d.h. er muß bei jedem Durchlauf eines Befehls erneut dieses reservierte Wort erkennen.

2.4.1.1 BASIC Befehlssatz

A Steuerbefehle

Diese Befehlsgruppe dient der Steuerung des Interpreters. Damit kann der Ablauf eines Programms, wie löschen, starten, verbessern, ausdrucken, bestimmt werden.

LIST
Ein zuvor eingegebenes Programm kann mit diesem Befehl ausgegeben werden. Dabei wird das gesamte Programm ausgegeben, wenn der Befehl LIST cr (Wagenrücklauf) eingegeben wird. Nach dem Befehl LIST kann auch eine zusätzliche Zahl angegeben werden, die die Zeilennummer angibt, von der an ein Programm gelistet werden soll. LIST 1ØØ cr gibt z.B. ein Programm, beginnend mit der Zeilennummer 1ØØ aus, falls diese vorhanden ist. Sonst wird die nächsthöhere Zeile gewählt. Mit CTRL C (Code 3) kann das Listing abgebrochen werden.

RUN
Nach Eingabe von RUN cr wird ein zuvor eingegebenes Programm gestartet, beginnend mit der niedrigsten Zeilennummer.

NEW
Damit kann ein Programm gelöscht werden. Anschließend ist es möglich, ein neues Programm einzugeben.

BYE
Bei der Ausführung dieses Befehls kehrt das BASIC System in den Monitor zurück. Das BASIC Programm bleibt erhalten, wenn anschließend auf die Adresse RSTART (hier 1ØØ3) gesprungen wird. Bei einem Sprung auf 1ØØØ würde das Programm gelöscht werden.

END
In der ursprünglichen Bedeutung steht dieser Befehl am Ende eines Programms. Da dies bei dem vorliegenden INTERPRETER nicht erforderlich ist, konnte der Befehl hier mit einer neuen Bedeutung versehen werden. Nach dem Starten des INTERPRETERs steht für Anwenderprogramme immer ein Speicherplatz von ungefähr 700 Byte zur Verfügung. Beim Überschreiten dieses Bereichs wird vom BASIC eine Fehlermeldung (SORRY) ausgegeben. Nun kann es aber sein, daß der Benutzer mehr Speicher zur Verfügung hat. Er kann es dem BASIC durch den jetzt so interpretierten Befehl END mitteilen, der dazu mit einem zusätzlichen Parameter versehen wird. Dieser Parameter stellt die absolute Adresse der gewünschten neuen Adresse für das BASIC Programm dar. Dabei muß berücksichtigt werden, daß der INTERPRETER noch etwa 140 Byte über diese Adresse hinaus benötigt. Um den INTERPRETER z.B. für einen Speicher von 32 KByte (Ø . . . 7FFFH) zu initialisieren, wird der Befehl END HEX(7FFF)-140 cr gegeben. Der Befehl HEX()!rechnet einen Sedezimalwert in einen Dezimalwert um.

B Programmierbare Befehle

Alle nun folgenden Befehle können im Gegensatz zu den Steuerbefehlen programmiert werden. Die meisten können auch im sogenannten „Direkt Modus“ verwendet werden,

d.h. einfach durch Eingabe ohne vorangestellte Zeilennummer. Sie werden dann unmittelbar nach der Eingabe von cr ausgeführt.

LET

Hier wird einer Variablen ein Wert zugewiesen.

Beispiel:
```
10 LET A=10
20 LET B= 2* (3-9) * 6/2
30 LET A=C
```

Dabei kann der Befehl LET auch weggelassen werden. Er dient eigentlich nur zur besseren Lesbarkeit.

FOR TO NEXT

Damit ist es möglich, Schleifen aufzubauen. Es wird dann eine bestimmte Befehlsfolge n-mal durchlaufen.

Beispiel:
```
10 FOR A=1 TO 10 STEP 2
20...
30...
40 NEXT A
```

Der Bereich zwischen 10 und 40 wird dabei 5mal durchlaufen. Innerhalb der Schleife kann der Wert von A verwendet werden, er sollte jedoch nicht verändert werden.

Die Angabe STEP legt die Schrittweite fest. Sie kann auch negativ sein. Es müssen dann aber auch der Anfangs- und Endwert entsprechend gewählt werden.

Beispiel:
```
10 FOR A=10 TO 1 STEP -2
20...
30...
40 NEXT A
```

Dieses Programm wirkt genau wie beim ersten Beispiel, nur daß hier die Variable A zunächst den Wert 10 erhält und dann den Wert 8 usw.. Wird die Angabe STEP weggelassen, so wird die voreingestellte Schrittweite von 1 angenommen.

GOTO

Der Befehl GOTO ist eine Sprunganweisung. Dabei wird nach dem Befehl GOTO eine Zeilennummer angegeben, auf die gesprungen werden soll. Diese Zahl kann auch ein arithmetischer Ausdruck sein. Damit ist es möglich, Mehrfachverzweigungen zu realisieren.

Beispiel:
```
10 A=10
20 GOTO A+10
```

Es wird zur Zeile 20 gesprungen und damit ein dynamischer Stop erreicht. Der Stop kann mit CTRL C unterbrochen werden. Es ist auch möglich, den Befehl GOTO im Direktmodus zu verwenden. Es wird dann ein im Speicher stehendes Programm von der angegebenen Zeilennummer an ausgeführt.

GOSUB

Mit dem Befehl GOSUB ist es möglich, einen Aufruf für ein Unterprogramm durchzuführen. Dabei wird wie beim Befehl GOTO eine Zeilennummer angegeben, die auch hier wieder berechnet werden kann. Das Unterprogramm endet mit dem Befehl RETURN.

RETURN
Der Befehl bewirkt einen Rücksprung in ein Hauptprogramm, das an die Stelle über einen Befehl GUSUB gekommen ist. Unterprogramme dürfen geschachtelt werden.

IF
Mit dieser Anweisung kann eine Entscheidung getroffen werden. IF wird von einem arithmetischen Ausdruck gefolgt. Ist der Wert ungleich 0, so wird der nachfolgende Befehl ausgeführt, andernfalls die nächste Zeile.

Beispiel: 1Ø IF A=2 GOTO 30
2Ø . . .

Wenn A den Wert A=2 hat, wird dieser Ausdruck logisch 1 (wahr) und ist damit ungleich 0 (unwahr). Der nächste Befehl wird ausgeführt. Andernfalls wird der Ausdruck 0 (unwahr) und die Zeile 20 wird ausgeführt. (THEN darf nicht verwendet werden).

REM
Die Anweisung REM ermöglicht es, Kommentare in das Programm BASIC einzufügen und damit die Übersichtlichkeit zu steigern. Dabei wird der Text, der nach REM steht, bis zum Zeilenende vom INTERPRETER ignoriert.

INPUT
Einer der wichtigsten Befehle ist INPUT. Er ermöglicht es, Daten im Dialogverfahren in das Programm einzugeben. Soll z.B. der Variablen C im Programm ein Wert zugewiesen werden, den der Benutzer erst nach dem Starten des Programms festlegt, so kann der Befehl folgendermaßen lauten:

Beispiel: 1Ø INPUT C

Bei der Ausführung des Programms druckt der INTERPRETER C:

Nun muß der Benutzer eine Zahl oder einen arithmetischen Ausdruck eingeben, der noch berechnet wird. Soll erreicht werden, daß anstatt des Namens der Variablen ein bestimmter Text ausgedruckt wird, so wird dieser Text von der Variablen in ein Anführungszeichen gesetzt.

Beispiel: 1Ø INPUT "Geben Sie eine Zahl ein" C

Bei der Ausführung des Programms wird dann der angegebene Text vor einem Doppelpunkt ausgegeben, und der Benutzer kann den Wert eingeben. Es ist auch möglich, mehrere Variablen einzugeben. Dazu werden Sie mit Kommas getrennt.

Beispiel: 1Ø INPUT A,C,'ZAHL'F,K,'WEITERER TEXT' W

PRINT
Mit Hilfe des Befehls PRINT ist es möglich, Daten und Texte auszugeben. Dazu werden die verschiedenen Variablen, Zahlen und Texte – jeweils durch Kommas getrennt – angegeben.

Beispiel: 2Ø PRINT 2,B,"TEXT",7+3

Dieses Programm bewirkt den Ausdruck der Zahl 2, dann den Inhalt der Variablen B, dann den Text TEXT und zum Schluß 10. Zahlen werden im Normalfall mit sechs Stellen ausgegeben. Dies kann aber geändert werden. Dazu dient eine spezielle Formatanweisung. Sie kann auch mehrmals in einer PRINT-Anweisung angegeben werden und bleibt dann bis zur nächsten Formatanweisung wirksam. Bei der Ausführung des nächsten Befehls PRINT ist wieder der Wert 6 vorangestellt. Die Formatanweisung

wird mit dem Zeichen # eingeleitet und hat als Parameter eine Zahl oder einen arithmetischen Ausdruck.

Beispiel: 10 PRINT 1,#10,1,1,#A,2

Die erste 1 wird mit insgesamt sechs Stellen ausgedruckt, die beiden anderen mit zehn Stellen und die 2 mit A-Stellen. Ist der Wert in der Formatanweisung kleiner, als die Anzahl der nötigen Stellen, so wird die gesamte Zahl ausgedruckt, doch ohne Leerzeichen. Wird bei der Anweisung PRINT an die letzte Stelle einer Zeile ein Komma gesetzt, so wird der Zeilenvorschub unterdrückt, und die nächste Anweisung PRINT gibt bei der letzten Position auf dieselbe Zeile aus.

STOP

Hierbei wird der Programmablauf beendet.

CALL

Mit dem Befehl CALL ist es möglich, Unterprogramme, die in Maschinensprache geschrieben sind, aufzurufen. Dafür ist ein Parameter anzugeben, der die absolute Adresse des Unterprogramms abgibt.

Beispiel: 1Ø Call HEX(54FE)

Dieses Programm bewirkt, daß das Maschinenprogramm auf der Adresse 54FEH aufgerufen wird. Das Unterprogramm muß mit einem RET (Code C9H) enden, dann wird in dem Programm BASIC wieder weitergefahren.

Mit PEEK und POKE können Parameter an das Unterprogramm über eine feste Adresse gegeben werden.

OUTCHAR

Mit diesem Befehl werden Einzelzeichen ausgegeben, die auch Sonderzeichen oder sonst nicht darstellbare Zeichen sein können. Dem Befehl wird als Parameter ein dezimaler Wert gegeben.

Beispiel: 1Ø OUTCHAR(65)

Bei der Ausführung wird das Zeichen A gedruckt.

OUT

Mit OUT kann ein Wert einem Port des 8080 (Z80) direkt zugewiesen werden. OUT wird dabei ähnlich wie eine Variable verwendet. Soll z.B. dem Port mit der Adresse 18H der Wert 2 zugewiesen werden, so ergibt der Befehl folgendes

Beispiel: 1Ø OUT(HEX(18))=2

Mit der Funktion HEX wird hier wieder erreicht, daß der sedezimale Wert 18 in einen dezimalen Wert umgerechnet und dann dem Befehl OUT zugeführt wird.

O$

Der Befehl O$ ist ein spezieller Befehl, der eingeführt wird, um auch hier Stringverarbeitung TINY BASIC durchführen zu können. Der Befehl ermöglicht es, einen Text auszugeben, der auf einer beliebigen Adresse steht und mit dem Wert 0 abschließt. Hierzu vergleiche auch die Befehle I$,PEEK,POKE. Dazu erhält der Befehl einen weiteren Wert, der die Anfangsadresse des Textes darstellt.

Beispiel: 10 O$ TOP

Hier wird ein Text ausgegeben, der auf der ersten freien Adresse nach dem Benutzerprogramm liegt. Dieser Text mußte natürlich zuvor dort gespeichert werden, z.B. mit POKE oder I$.

I$

Dieser Befehl ist das Gegenstück zum Befehl O$. Diese Anweisung erhält als zusätzlichen Parameter die Anfangsadresse, auf die ein Text abgelegt werden soll. Die Eingabe eines Textes wird durch cr beendet. Das Ende des Textes wird mit 0 gekennzeichnet. Die Länge des eingegebenen Textes ist mit LEN feststellbar.

Beispiel: 10 I$ TOP

Dieses Programm legt einen Text, beginnend auf der ersten freien Adresse, ab.

POKE

POKE ist ein Befehl, mit dem auf einen Speicher ein direkter Zugriff erfolgen kann. Dabei besteht automatisch ein Schreibschutz für ein abgelegtes Programm BASIC. POKE besitzt zwei Parameter. Der erste gibt die absolute Adresse an, der zweite bestimmt den dezimalen Wert von 0 bis 255, der auf diese Adresse abgelegt werden soll.

Beispiel: 1Ø POKE TOP +1,5
2Ø POKE 16ØØØ,2*5
3Ø POKE TOP,'T'
4Ø POKE HEX(2ØØØ),A

Die Abarbeitung erfolgt so:

Zeile 10: Der Wert 5 wird auf die zweite freie Speicherzelle gelegt.
Zeile 20: Der Wert 10 wird auf die Adresse 16ØØØD gelegt.
Zeile 30: Der 7-Bit-Code (ASCII) des Zeichens T kommt auf die erste freie Speicherzelle.
Zeile 40: Auf die Adresse 2ØØØH wird der Inhalt von A (untere 8 Bit) gelegt.

HEX

Dem Befehl HEX wird in Klammern ein sedezimaler Wert zugefügt, der dann in einen dezimalen Wert umgerechnet wird.

IN

Mit IN kann der Wert eine Port des 8080 geladen werden.

Beispiel: 1Ø A=IN(HEX(18))

Hier wird der Variablen A der Wert des Port mit der Adresse 18H zugewiesen.

TOP

TOP ist eine Pseudovariable, d.h. es kann ihr kein Wert zugewiesen werden, sondern es wird immer nur ein Wert von ihr geliefert. Mit dieser Funktion läßt sich die Adresse des ersten freien Speicherplatzes ermitteln. Vor dieser Speicherzelle steht das BASIC–Anwenderprogramm.

LEN

LEN ist ebenfalls eine Pseudovariable. Ihr Wert gibt die Länge des zuletzt mit I$ eingegebenen Textes an.

BYTE
Der Befehl gibt eine als Parameter angegebene Zahl in sedezimaler Schreibweise aus.
Beispiel: 1Ø BYTE(1Ø)
Es wird eine Folge ØA ausgegeben.

WORD
Dieser Befehl wirkt wie der vorhergehende, nur daß ein Wert mit vier Stellen ausgegeben wird.
Beispiel: 1Ø WORD(1Ø)
Es wird ØØØA ausgegeben.

TAB
Mit TAB kann die Schreibposition verändert werden. Dabei wird im Gegensatz zum Standard-BASIC der Befehl TAB nicht in eine Anweisung PRINT geschrieben.
Beispiel: 1Ø TAB(2Ø)
Die Schreibposition wird um 20 Leerzeichen weitergerückt.

RND
Die Funktion RND liefert einen Zufallswert. Dabei kann noch angegeben werden, in welchem Bereich dieser Wert liegen muß.
Beispiel: 1Ø A=RND(1ØØØ)
Die Variable A erhält einen Wert zwischen 1 und 1ØØØ.

ABS
ABS bildet den Absolutbetrag einer Zahl.
Beispiel: ABS(-2)
Hier wird der Betrag 2 gebildet.

SIZE
Mit SIZE kann der für das Benutzerprogramm noch zur Verfügung stehende Raum berechnet werden.

PEEK
Mit PEEK kann ein Zugriff auf einen Speicher direkt durchgeführt werden. Dazu wird eine Absolutadresse angegeben.
Beispiel: 1Ø A = PEEK(HEX(2ØØØ))
A erhält den Wert des Byte, welches an der Adresse 2ØØØ sedezimal steht.

INCHAR
Mit dem Befehl INCHAR kann ein Zeichen von der Konsole geholt werden. Dabei wird das Zeichen nicht ausgegeben. Dies ermöglicht es, Zeichen umzudefinieren oder Steuertasten zu definieren.
Beispiel: 1Ø B = INCHAR
In die Variable B wird der entsprechende 7-Bit-Code (ASCII) geladen, der dem von der Konsole eingegebenen Zeichen entspricht.

C Weitere Eigenschaften

Variable
Als Variable stehen A bis Z zur Verfügung. Als Array (Dimension 1) wird das Zeichen @ verwendet.
Beispiel: 1Ø@(10)=6

Die maximale Größe des Arrays hängt dabei von der Länge des Anwenderprogramms und dem gesamten Arbeitsspeicher ab. Das Array kann wie die Variablen pro Element 2 Byte fassen (± 32767).

Arithmetik
Der Zahlenbereich reicht von -32768 bis +32767. Zugelassen sind die vier Grundrechenarten mit den Zeichen + - * / . Klammern können beliebig gesetzt und verschachtelt werden.

Logische Operatoren
< >< = > = #= liefern den logischen Wert 1, falls die Aussage wahr ist und den Wert 0, falls nicht. Die Operatoren können beliebig mit den Zeichen + - * und / verknüpft werden.

Textoperator
Mit ' ' kann der dezimale Code eines beliebigen Zeichens (nach DIN 66003, ISO, bzw. 7-Bit-Code) ermittelt werden. 'A' liefert z.B. den Wert 65.

Steuerzeichen
Mit verschiedenen Steuerzeichen kann die Eingabe, sowie der Ablauf eines Programms kontrolliert werden. CTRL C (Code Ø3H), d.h.die Tasten CTRL und C werden auf der Tastatur gleichzeitig betätigt), z.B. unterbricht die Ausführung eines gerade laufenden Programms oder Listings. Mit CTRL A (Code Ø1H) wird ein zuletzt eingegebenes Zeichen gelöscht. Mit ESC (Code 1BH) wird die gerade eingegebene Zeile gelöscht, wenn cr noch nicht gegeben wurde. CTRL B besitzt eine besondere Bedeutung (Code Ø2H). Wird CTRL B ausgeführt, so gibt der INTERPRETER keine Zeichen mehr aus, aber empfängt noch alle Zeichen. Damit ist es möglich, z.B. Programme aus einem Cassettenrecorder zu laden. Die Programme werden mit Hilfe des Befehls LIST ausgegeben, während der Cassettenrecorder läuft und der Routine der Konsole parallelgeschaltet ist. Bei der Wiedergabe wird zunächst NEW und dann CTRL B eingegeben. Dann wird der Cassettenrecorder gestartet, der durch Software oder Hardware der Eingaberoutine parallelgeschaltet ist. Das Programm kann dann eingelesen werden. Am Schluß wird wieder CTRL B eingegeben, so daß der INTERPRETER wieder normal weiterarbeitet. Die Unterdrückung der Ausgabe ist nötig, weil sonst durch die Ausgabevorgänge eine Verlangsamung der Eingabevorgänge die Folge wäre und Zeichen überlesen würden.
Abb. 2.4.1.1-1 zeigt einige Programmbeispiele.

```
READY
>LIST
  10 REM STRINGVERARBEITUNG
  20 PRINT 'GEBEN SIE EINEN STRING EIN'
  30 I$ TOP;REM STRING WIRD EINGELESEN
  40 FOR I=1 TO LEN
  50 IF PEEK(TOP+I)=' ' POKE TOP+I,'-';REM UMWANDLUNG
  60 NEXT I
  70 O$ TOP;REM AUSGABE DES UMGEWANDELTEN STRING


  READY
  >RUN
  GEBEN SIE EINEN STRING EIN
  DIES IST EIN STRING MIT LEERZEICHEN
  DIES-IST-EIN-STRING-MIT-LEERZEICHEN
  READY
  >


  >LIST
    10 REM ARRAYVERARBEITUNG
    20 FOR I=1 TO 10
    30 FOR J=1 TO 10
    40 @(I)=0 ; REM LOESCHEN
    50 NEXT J
    60 NEXT I
    70 FOR I=1 TO 10
    80 FOR J=1 TO 10
    90 @(I)=@(I)+RND(100)
   100 NEXT J
   110 NEXT I
   120 REM AUSGEBEN
   130 FOR I=1 TO 10
   140 PRINT #5,@(I),
   150 NEXT I
   160 PRINT

  READY
  >RUN
    432  530  601  521  596  547  413  331  516  572

  READY
  >RUN
    565  494  657  598  571  537  640  340  592  540

  READY
  >RUN
    591  465  422  459  516  462  438  634  504  413
```

Abb. 2.4.1.1-1 BASIC-Programmbeispiele mit RDK BASIC

```
READY
>TAB(15);PRINT #4,A   ,B   ,A<B,A>B,A=B,A#B,A<=B,A>=B
                  4    5    1    0    0    1    1    0

READY
>PRINT (1=1)*(A<B)*100+A+A*('1'='1')
   108



READY
>LIST
  10 REM AUSDRUCKEN VON ZUFALLSZAHLEN ZWISCHEN 0 UND 9
  20 FOR I=0 TO 9
  30 PRINT RND(10)-1
  40 NEXT I
  50 PRINT
                                                    zu Abb. 2.4.1.1-1
READY
>RUN
      6
      5
      6
      4
      2
      0
      5
      0
      7
      4
READY
>PRINT SIZE
   500

READY
>END HEX(3FFF)-140

READY
>PRINT SIZE
  8918

READY
>PRINT TOP
  7193

READY
>WORD(TOP)
1C19
READY
>NEW

READY
>10 REM FEHLERBEHANDLUNG
>20 PRINT 2+3,34+A,B,1/0,3,4
>RUN
     5     38      SHOW?
  20 PRINT 2+3,34+A,B,1/0?,3,4

READY
>NEW
```

```
>LIST
  10 REM ANWENDUNG VON INCHAR UND OUTCHAR
  20 A=INCHAR
  30 IF A=' ' STOP
  40 IF A='0' B='*'
  50 IF A='1' B=' '
  60 OUTCHAR(B)
  70 GOTO 20

READY
>RUN
** * * *   * *   *   *
READY
>
```

zu Abb. 2.4.1.1-1

```
READY
>LIST
  10 REM TAB FUNKTION
  20 FOR I=-8 TO 8
  30 TAB(I*I)
  40 PRINT "*"
  50 NEXT I

READY
>RUN
        *
      *
    *
   *
    *
      *
           *
                *
>LIST
  10 REM MEMORY DUMP PROGRAMM
  20 INPUT 'ANFANGSADRESSE'A,'ENDADRESSE'E
  30 FOR I=A TO E STEP 8
  40 WORD(I);PRINT" :",
  50 FOR J=0 TO 7
  60 BYTE(PEEK(I+J));PRINT " ",
  70 NEXT J
  80 PRINT.
  90 NEXT I
 100 PRINT

READY
>

READY
>RUN
ANFANGSADRESSE:HEX(100)
ENDADRESSE:HEX(13F)
0100 :31 73 1B 3E C9 32 00 10
0108 :CD 00 10 3B 3B D1 21 12
0110 :00 19 11 00 10 01 9E 0A
0118 :ED B0 C3 00 10 C3 0F 10
0120 :C3 D5 10 C3 03 F0 C3 00
0128 :60 C3 12 F0 31 73 1B 3F
0130 :FF C3 D9 16 E3 CD 37 10
0138 :BE C3 7A 10 3E 0D C5 F5

READY
>
```

zu Abb. 2.4.1.1-1

```
-DFE3 1A8F

0FE3 31 F1 20 3E C9 32 00 10 CD 60 10 3B 3B 1. >.2.....;;
0FF0 D1 21 12 00 19 11 00 10 01 9F 0A ED B0 C3 00 10 .!..............
1000 C3 0F 10 C3 D5 10 C3 03 F0 C3 09 F0 C3 12 F0 31 ...............1
1010 F1 20 3E FF C3 09 16 E3 CD 37 10 BE C3 7A 10 3E . >.......7...z.>
1020 0D C5 F5 3A 06 20 B7 C3 0D 17 CD C4 13 E5 C3 86 ...:. ..........
1030 13 7C BA C0 7D BB C9 1A FE 20 C0 13 C3 37 10 F1 .|..}.... ...7..
1040 CD 27 15 C3 46 15 CD 37 16 D6 40 D6 C2 6A 16 13 .'..F..7..@..j..
1050 CD 7B 14 29 DA B3 10 D5 EB CD C2 14 CD 31 10 DA .{.).........1..
1060 70 15 2A 2A 21 CD E5 14 D1 C9 FE 1B 3F D8 13 21 p.**!.......?..!
1070 F3 20 07 85 6F 3E 00 8C 67 C9 23 CA 85 10 C5 4E . ..o>..g.#....N
1080 06 00 09 C1 1B 15 23 E3 C9 21 00 00 44 CD 37 16 ......#..!..D.7.
1090 FE 30 D8 FE 3A D0 3E F0 A4 C2 B3 10 04 C5 44 4D .0..:.>.......DM
10A0 29 29 09 29 1A 13 E6 0F 85 6F 3E 00 8C 67 C1 1A ))).....o>..g..
10B0 F2 90 10 D5 11 BA 10 C3 44 15 48 4F 57 3F 0D 0A ........D.HOW?..
10C0 52 45 41 44 59 0D 0A 57 48 41 54 3F 0D 0A 53 4F READY..WHAT?..SO
10D0 52 52 59 0D 0A 31 F1 20 CD 1F 10 11 C0 10 97 CD RRY..1. ........
10E0 E3 15 21 E9 10 22 07 20 21 00 00 22 0F 20 22 09 ..!..". !..". ".
10F0 20 3E 3E CD 76 15 D5 CD B5 18 CD 89 10 CD 37 10  >>.v.........7.
1100 7C B5 C1 CA 82 18 1B 7C 12 1B 7D 12 C5 D5 79 93 |......|..}...y.
1110 F5 CD B9 15 D5 C2 28 11 D5 CD D7 15 C1 2A 18 20 ......(......*.
1120 CD 7A 16 60 69 22 1B 20 C1 2A 1B 20 F1 E5 FE 63 .z.`i". .*. ....
1130 CA D5 10 85 6F 3E 00 8C 67 CD BD 18 CD 31 10 D2 ....o>..g....1..
1140 6F 15 22 1B 20 01 CD 85 16 D1 E1 CD 7A 16 C3 F1 o.". ........z...
1150 10 CD 3A 15 21 30 21 22 1B 20 CD 3A 15 C3 D5 10 ..:.!0!". .:....
1160 CD 3A 15 11 30 21 21 60 00 CD C1 15 DA D5 16 EB .:..0!!`........
1170 22 07 20 EB 13 13 CD 75 1A 21 /E 17 C3 85 18 CD ". ....u.!~......
1180 2A 10 D5 CD 3A 15 CD B9 15 C2 B4 10 F1 C3 6F 11 *...:.........o.
1190 CD 89 10 CD 3A 15 CD B9 15 DA D5 10 CD 65 16 CD ....:.........e..
11A0 75 1A CD C1 15 C3 99 11 0E 06 CD 17 10 3B 66 CD u............;f..
11B0 1F 10 C3 76 11 CD 17 10 0D 06 CD 1F 10 C3 66 11 ...v..........f.
11C0 CD 17 10 23 07 CD 2A 10 4D C3 D2 11 CD F1 15 C3 ...#..*.M.......
11D0 E3 11 CD 17 10 2C 06 CD 27 15 C3 C0 11 CD 1F 10 .....,..'.......
11E0 CD 3F 10 CD 2A 10 C5 CD 21 16 C1 C3 D2 11 CD B6 .?..*...!.......
11F0 16 CD 2A 10 D5 CD B9 15 C2 B4 10 2A 07 20 E5 2A ..*........*. .*
1200 09 20 E5 21 00 00 22 0F 20 39 22 09 20 C3 6F 11 . .!..". 9". .o.
1210 CD 3A 15 2A 09 20 7C B5 CA 40 15 F9 E1 22 09 20 .:.*. |..@...". 
1220 E1 22 07 20 D1 CD 94 16 CD 3F 10 CD B6 16 CD 0E .". .....?.......
1230 15 2B 22 0F 20 21 52 18 C3 85 18 CD 2A 10 22 13 .+". !R......*.".
1240 20 21 5A 16 C3 85 18 CD 2A 10 C3 50 12 21 01 00  !Z.....*...P.!..
1250 22 11 20 2A 07 20 22 15 20 EB 22 17 20 01 0A 00 ". *. ". . ". ....
1260 2A 0F 20 EB 60 68 39 3E 09 7E 23 B6 CA 89 12 7E *. .`h9>.~#....~
1270 2B BA C2 68 12 7E BB C2 68 12 EB 21 00 00 39 44 +..h.~..h..!..9D
1280 4D 21 0A 00 19 CD 85 16 F9 2A 17 20 EB CD 3F 10 M!.......*. ..?.
1290 CD 46 10 DA 40 15 22 0B 20 D5 EB 2A 0F 20 7C B5 .F..@.". ..*. |.
12A0 CA 41 15 CD 31 10 CA B3 12 D1 CD 94 16 2A 0B 20 .A..1........*. 
12B0 C3 99 12 5E 23 56 2A 11 20 E5 7C AA 7A 19 FA C5 ...^#V*. .|.z...
12C0 12 AC FA E9 12 EB 2A 0F 20 73 23 72 2A 13 20 F1 ......*. s#r*. .
12D0 B7 F2 D5 12 EB CD 94 15 D1 DA EB 12 2A 15 20 22 ............*. "
12E0 07 20 2A 17 20 EB CD 3F 10 E1 D1 CD 94 16 CD 3F . *. ..?.......?
12F0 10 21 00 00 C3 FA 12 CD 2A 10 7C B5 C2 76 11 CD .!......*.|..v..
1300 D9 15 D2 6F 11 C3 D5 10 2A 0D 20 F9 E1 22 07 20 ...o....*. ..". 
1310 D1 D1 D5 CD F1 15 C3 22 13 CD 46 10 DA 60 13 C3 .......".F..`..
1320 34 13 D5 CD 46 10 DA 40 15 1A 4F 97 12 D1 CD E3 4...F..@..O.....
1330 15 79 18 12 05 EB 2A 07 20 E5 21 12 13 22 07 20 .y....*. .!..". 
1340 21 00 00 39 22 0D 20 D5 3E 3A CD 76 15 CD B5 18 !..9". .>:.v....
1350 CD 2A 10 CD 75 1A D1 EB 73 23 72 E1 22 07 20 D1 .*..u...s#r.". .
1360 F1 CD 17 10 2C 03 C3 12 13 CD 3F 10 1A FE 0D CA ....,.....?.....
1370 7D 13 CD 0E 15 CD 17 10 2C 03 C3 72 13 CD 3F 10 }.......,..r..?.
1380 21 64 18 C3 85 18 CD AF 13 D8 6F C9 CD AF 13 C8 !d........o.....
1390 6F C9 CD AF 13 C8 D8 6F C9 CD AF 13 6F C8 D8 6C o......o....o..l
13A0 C9 CD AF 13 C0 6F C9 CD AF 13 D0 6F C9 E1 C9 79 .....o.....o...y
13B0 E1 C1 E5 C5 4F CD C4 13 EB E3 CD 04 15 01 21 00 ....O.........!.
13C0 00 3E 01 C9 CD 17 10 2D 06 21 00 00 C3 F6 13 CD .>.....-.!......
13D0 17 10 2B 00 CD 00 14 CD 17 10 2B 15 E5 CD 00 14 ..+.......+.....
13E0 EB E3 7C AA 7A 19 D1 FA D7 13 AC F2 D7 13 C3 B3 ..|.z...........
13F0 10 CD 17 10 2D 92 E5 CD 00 14 CD EF 14 C3 E8 13 ....-...........
```

Abb. 2.4.1.2-1 Objectcode des BASIC-Interpreters

```
1400 CD 64 14 CD 17 10 2A 2D E5 CD 64 14 06 00 CD EC .d....*-..d.....
1410 14 E3 CD EC 14 EB E3 7C B7 CA 22 14 7A B2 E8 C2 .......|..".z...
1420 B4 10 7D 21 00 00 B7 CA 56 14 19 DA B4 10 3D C2 ..}!....V.....=.
1430 2A 14 C3 56 14 CD 17 10 2F 4E E5 CD 64 14 06 00 *..V..../N..d...
1440 CD EC 14 E3 CD EC 14 EB E3 EB 7A B3 CA B4 10 C5 ..........z.....
1450 CD CF 14 60 69 C1 D1 7C B7 FA B3 10 78 B7 FC EF ...`i..|....x...
1460 14 C3 03 14 21 0A 18 C3 85 18 CD 46 10 DA 75 14 ....!......F..u.
1470 7E 23 66 6F C9 CD 89 10 78 B7 C0 CD 17 10 28 09 ~#fo....x.....(.
1480 CD 2A 10 CD 17 10 29 01 C9 C3 40 15 CD 7B 14 7C .*....)...@..{.|
1490 B7 FA B3 10 85 CA B3 10 D5 E5 2A 19 20 11 05 20 ..........*. ..
14A0 CD 31 10 DA A9 14 21 0F 10 5E 23 56 22 19 20 E1 .1....!..^#V". .
14B0 E8 C5 CD CF 14 C1 D1 23 C9 CD 7B 14 1B CD EC 14 .......#..{.....
14C0 13 C9 2A 1B 20 D5 EB 2A 2A 21 CD E5 14 D1 C9 E5 ..*. ..**!......
14D0 6C 26 00 CD DA 14 41 70 E1 67 0E FF 0C CD E5 14 l&....Ap.g......
14E0 D2 DC 14 19 C9 7D 93 6F 7C 9A 67 C9 7C B7 F0 7C .....}.o|.g.|..|
14F0 B5 C8 7C F5 2F 67 7D 2F 6F 23 F1 AC F2 B3 10 78 ..|./g}/o#.....x
1500 EE 80 47 C9 7C AA F2 0A 15 EB CD 31 10 C9 CD 46 ..G.|......1...F
1510 10 DA 40 15 E5 CD 17 10 3D 0A CD 2A 10 44 4D E1 ..@.....=..*.DM.
1520 71 23 70 C9 C3 40 15 CD 17 10 3B 04 F1 C3 76 11 q#p..@....;...v.
1530 CD 17 10 0D 94 F1 C3 66 11 C9 CD 37 10 FE 00 C8 .......f...7....
1540 D5 11 C7 10 97 CD E3 15 D1 1A F5 97 12 2A 67 20 .............*g
1550 E5 7E 23 B6 D1 CA D5 10 7E B7 FA 08 13 CD 65 16 .~#.....~.....e.
1560 1B F1 12 3E 3F CD 21 10 97 CD E3 15 C3 D5 10 D5 ...>?.!.........
1570 11 CE 10 C3 44 15 CD 21 10 CD B5 18 CD 2C 17 FE ....D..!.....,..
1580 01 CA A1 15 CD 21 10 FE 0A CA 7C 15 B7 CA 7C 15 .....!....|...|.
1590 FE 1B CA B1 15 12 13 FE 0D C8 7B CD C5 18 C2 7C ..........{....|
15A0 15 7B CD CC 18 CA B1 15 1B 3E 08 CD 21 10 C3 7C .{.......>..!..|
15B0 15 CD 1F 10 3E 08 C3 76 15 7C B7 FA B3 10 11 30 ....>..v.|.....0
15C0 21 E5 2A 1B 20 2B CD 31 16 E1 D8 1A 95 47 13 1A !.*. +.1.....G..
15D0 9C DA D8 15 1B B0 C9 13 13 1A FE 0D C2 D8 15 13 ................
15E0 C3 C1 15 47 1A 13 B8 C8 CD 21 10 FE 0D C2 E4 15 ...G.....!......
15F0 C9 CD 17 10 22 0F 3E 22 CD E3 15 FE 0D E1 CA 66 ....".>".......f
1600 11 23 23 23 E9 CD 17 10 27 05 3E 27 C3 F6 15 CD .###....'.>'....
1610 17 10 5F 0C 3E 8D CD 21 10 CD 21 10 E1 C3 01 16 .._.>..!..!.....
1620 C9 06 00 CD EC 14 F2 2C 16 06 2D 0D D5 11 0A 00 .......,..-.....
1630 D5 0D C5 CD CF 14 78 B1 CA 43 16 E3 2D E5 60 69 ......x..C..-.`i
1640 C3 33 16 C1 0D 79 B7 FA 52 16 3E 20 CD 21 10 C3 .3...y..R.> .!..
1650 44 16 78 B7 C4 21 10 5D 7B FE 0A D1 C8 C6 30 CD D.x..!.]{.....0.
1660 21 10 C3 58 16 1A 6F 13 1A 67 13 0E 04 CD 21 16 !..X..o..g....!.
1670 3E 20 CD 21 10 97 CD E3 15 C9 CD 31 10 C8 1A 02 > .!.......1....
1680 13 03 C3 7A 16 78 92 C2 8D 16 79 93 C8 1B 2B 1A ...z.x....y...+.
1690 77 C3 85 16 C1 E1 22 0F 20 7C B5 CA AE 16 E1 22 w.....". |......"
16A0 11 20 E1 22 13 20 E1 22 15 20 E1 22 17 20 C5 C9 . .". . .". . .". ..
16B0 21 27 20 CD FF 14 C1 39 D2 6F 15 2A 0F 20 7C B5 !' ....9.o.*. |.
16C0 CA D6 16 2A 17 20 E5 2A 15 20 E5 2A 13 20 E5 2A ...*. .*. .*. .*
16D0 11 20 E5 2A 0F 20 E5 C5 C9 32 06 20 16 03 CD 1F . .*. ...2. ....
16E0 10 15 C2 DF 16 97 11 46 17 CD E3 15 21 0F 10 22 .......F....!..."
16F0 19 20 21 30 21 22 10 20 21 8A 23 22 2A 21 21 8C . !0!". !.#"*!!.
1700 23 22 2C 21 21 CC 23 22 2E 21 C3 D5 10 C2 13 17 #",!!.#".!......
1710 F1 C1 C9 F1 F5 4F 79 FE 0D CA 22 17 CD 09 10 F1 .....Oy..."..... 
1720 C1 C9 0E 0D CD 09 10 0E 0A C3 1C 17 CD 66 10 E6 .............f..
1730 7F FE 02 C2 40 17 3A 06 20 2F 32 06 20 C3 2C 17 ....@.:. /2. .,.
1740 FE 03 C0 C3 D5 10 52 44 4B 20 50 52 4F 4D 50 54 ......RDK PROMPT
1750 20 42 41 53 49 43 20 56 33 2E 32 20 33 4B 0D 0A  BASIC V3.2 3K..
1760 4C 49 53 54 00 90 11 52 55 4E 00 60 11 4E 45 57 LIST...RUN.`.NEW
1770 00 51 11 42 59 45 00 04 19 45 4E 44 00 D3 18 4E .Q.BYE...END...N
1780 45 58 54 00 90 12 4C 45 54 00 72 13 49 46 00 F7 EXT...LET.r.IF..
1790 12 47 4F 54 4F 00 7F 11 47 4F 53 55 42 00 EE 11 .GOTO...GOSUB...
17A0 52 45 54 55 52 4E 00 18 12 52 45 4D 00 F1 12 46 RETURN...REM...F
17B0 4F 52 00 2B 12 49 4E 50 35 34 00 12 13 50 52 49 OR.+.INPUT...PRI
17C0 4E 54 00 A8 11 53 54 4F 50 00 5A 11 43 41 4C 4C NT...STOP.Z.CALL
17D0 00 08 19 4F 55 54 43 48 41 52 00 1A 1A 4F 55 54 ...OUTCHAR...OUT
17E0 00 15 19 4F 24 00 66 19 49 24 00 73 19 50 4F 4B ...O$.f.I$.s.POK
17F0 45 00 AC 19 54 41 42 00 3B 19 42 59 54 45 00 CE E...TAB.;.BYTE..
```

zu Abb. 2.4.1.2-1

2.4.1.2 Starten und Anpassen des Interpreters

Abb. 2.4.1.2-1 zeigt den Objektcode des INTERPRETERS, *Abb. 2.4.1.2-2* das Programm im INTEL-HEX-Format. Das Programm startet auf der Adresse 1ØØØH und ist etwa 3 KByte lang. Das Programm kann in ROM-Speichern abgelegt werden. Auf Adresse 2ØØØH beginnt der RAM-Speicher. Variable, die von INTERPRETERN benötigt werden, werden beginnend mit dieser Adresse, abgelegt. Um den INTERPRETER zu starten, ist mindestens ein Platz von 1 KByte im RAM erforderlich. *Abb. 2.4.1.2-3* zeigt den Anfang des Programmlistings. Das Programm startet mit einem LOADER, der die Aufgabe hat, den INTERPRETER aus einem ROM-Bereich in ein RAM zu verschieben. Das ist erforderlich, wenn z.B. der gesamte untere Speicher mit RAM belegt ist. Dazu wird auf die Zieladresse 1ØØØH ein Befehl C9H geschrieben, um damit anschließend die Lage des ROM festzustellen. Dann wird der INTERPRETER in diesen Bereich verschoben. Das Verschieben geschieht mit einem

```
1800 19 57 4F 52 44 00 D8 19 00 6C 13 52 4E 44 00 8C .WORD....l.RND..
1810 14 41 42 53 00 B9 14 53 49 5A 45 00 C2 14 50 45 .ABS...SIZE...PE
1820 45 4B 00 A5 19 49 4E 43 48 41 52 00 24 1A 48 45 EK...INCHAR.$.HE
1830 58 00 2B 1A 49 4E 00 4C 19 27 90 02 1A 54 4F 50 X.+.IN.L.'...TOP
1840 00 10 1A 4C 45 4E 00 15 1A 43 53 54 53 00 FA 19 ...LEN...CSTS...
1850 00 6A 14 54 4F 00 3B 12 00 40 15 53 54 45 50 00 .j.TO.;..@.STEP.
1860 47 12 00 4D 12 3E 3D 00 86 13 23 66 8C 13 3E 00 G..M.>=...#...>.
1870 92 13 3D 00 A1 13 3C 3D 00 99 13 3C 00 A7 13 00 ..=...<=...<....
1880 AD 13 21 5F 17 CD 37 10 D5 1A 13 FE 2E CA A7 18 ..!_..7.........
1890 23 BE CA 89 18 3E 00 1B BE CA AE 18 23 BE C2 9C #....>......#...
18A0 18 23 23 D1 C3 85 18 3E 00 23 BE C2 A9 18 23 7E .##....>.#....#~
18B0 23 66 6F F1 E9 E5 2A 2C 21 54 5D E1 C9 E5 2A 2A #fo...*,!T]...**
18C0 21 54 5D E1 C9 E5 2A 2E 21 BD E1 C9 E5 2A 2C 21 !T]...*.!....*,!
18D0 BD E1 C9 CD 2A 10 EB 21 8A 23 EB CD 31 10 DA 70 ....*..!.#..1..p
18E0 15 7C B7 FA 70 15 7E 2F 77 46 B8 C2 70 15 22 2E .|..p.~/wF..p."."
18F0 21 7D D6 84 6F 7C DE 00 67 22 2C 21 2B 2B 22 2A !}..o|..g",!++"*
1900 21 C3 D5 10 FF C3 D5 10 CD 2A 10 D5 61 11 19 C5 !........*.....
1910 E9 D1 CD 3F 10 CD 7B 14 E5 CD 17 10 3D 1A CD 2A ...?..{.....=..*
1920 10 45 3E D2 32 02 26 E1 7D 32 03 26 3E C9 32 04 .E>.2. .}2. >.2.
1930 20 78 CD 02 20 CD 3F 10 C3 40 15 CD 7B 14 7C B5  x.. .?..@..{.|.
1940 CC 3F 10 2B 3E 20 CD 21 10 C3 3E 19 CD 7B 14 E5 .?.+> .!..>..{..
1950 3E DB 32 02 20 E1 7D 32 03 20 3E C9 32 04 20 CD >.2. .}2. >.2. .
1960 02 20 26 00 6F C9 CD 2A 10 D5 EB AF CD E3 15 D1 . &.o..*........
1970 CD 3F 10 CD 2A 10 D5 EB 2A 18 20 EB CD 31 10 DA .?..*...*. ..1..
1980 70 15 CD B5 18 CD 7C 15 44 4D EB 2B CD B5 18 D5 p.....|.DM.+....
1990 CD 7A 16 AF 92 D1 23 CD E5 14 EB 21 00 20 73 23 .z....#....!. s#
19A0 72 D1 CD 3F 10 CD 7B 14 6E 26 00 C9 CD 2A 10 D5 r..?..{.n&...*..
19B0 EB 2A 18 20 EB CD 31 10 DA 70 15 D1 E5 CD 17 10 .*. ..1..p......
19C0 2C 09 CD 2A 10 7D E1 77 CD 3F 10 C3 40 15 CD 7B ,..*.}.w.?..@..{
19D0 14 7D CD E6 19 CD 3F 10 CD 7B 14 7C CD E6 19 7D .}....?..{.|...}
19E0 CD E6 19 CD 3F 10 F5 0F 0F 0F 0F CD EF 19 F1 E6 ....?...........
19F0 0F C6 90 27 CE 40 27 C3 21 10 CD 0C 10 2F 6F 26 ...'.@'.!..../o&
1A00 00 C9 1A 13 6F 26 00 CD 17 10 27 61 C9 C3 40 15 ....o&....'a..@.
1A10 2A 18 20 23 C9 2A 00 20 2B C9 CD 2A 10 7D CD 21 *. #.*. +..*.}.!
1A20 10 CD 3F 10 CD 2C 17 26 00 6F C9 C5 21 00 00 CD ..?..,.&.o..!...
1A30 17 10 28 10 1A FE 0D CA 40 15 CD 56 1A 29 29 29 ..(.....@..V.)))
1A40 29 06 00 4F 09 13 CD 17 10 29 03 C3 54 1A C3 34 )..O.....)..T..4
1A50 1A C3 40 15 C1 C9 FE 30 FA 40 15 FE 39 FA 60 1A ..@....0.@..9.m.
1A60 CA 6D 1A FE 41 FA 40 15 FE 47 F2 40 15 D6 36 FE .m..A.@..G.@..0.
1A70 0A F8 D6 07 C9 CD 0C 10 C8 CD 06 10 FE 03 C8 C3 ................
1A80 D5 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

zu Abb. 2.4.1.2-1

```
!.MAIN.7F
\03.PROG.010000.DATA.020000.BLNK.0300006A
:03010000C3E30F47
:180FE30031F1203EC9320010CD00103B3BD121120019110010019F9A30
:180FFB00EDB0C30010C30F10C3D510C303F0C309F0C312F031F1203E2D
:18101300FFC3D916E3CD3710BEC37A103E0DC5F53A0620B7C30D17CD42
:18102B06C413E5C380137CBAC07DBBC91AFE20C013C33710F1CD271595
:18104300C34015CD3710D640D8C26A1913CD7B1429DAB310D5EBCDC2BB
:18105B0014CD3110DA70152A2A21CDE514D1C9FE1B3FD81321F32007A9
:18107300856F3E008C67C923CA8510C54E060009C11B1323E3C92100F4
:18108B000044CD3710FE30D8FE3AD03EF6A4C2B31604C5444D292909DB
:1810A300291A13E60F856F3E008C67C11AF29010D511BA10C344154844
:1810BB004F573F0D0A52454144590D0A574841543F0D0A534F5252596C
:1819D3000D0A31F120CD1F1011C01097CDE31521E910220720210000EF
:1810EB00220F202209203E3ECD7615D5CDB518CD8910CD37107CB5C1A2
:18110300CA82181B7C121B7D12C5D57993F5CDB915D5C22811D5CDD79E
:18111B0015C12A1B20CD7A166069221B20C12A1B20F1E5FE03CAD51052
:18113300856F3E008C67CD8D18CD3110D26F15221B20D1CD8516D1E131
:18114B00CD7A16C3F110CD3A15213021221B20CD3A15C3D516CD3A15A0
:18116300113021210000CDC115DAD510EB220720EB1313CD751A217E4F
:18117B0017C38518CD2A10D5CD3A15CDB915C2B416F1C36F11CD891032
:18119300CD3A15CDB915DAD510CD6516CD751ACDC115C399110E06CD39
:1811AB0017103B06CD1F10C37611CD17100D06CD1F10C36611CD17104D
:1811C3002307CD2A104DC3D211CDF115C3E311CD17102C06CD2715C374
:1811DB00C011CD1F10CD3F10CD2A10C5CD2116C1C3D211CDB016CD2A52
:1811F30010D5CDB915C2B4102A0720E52A0920E5210000220F203922A3
:18120B000920C36F11CD3A152A09207CB5CA4615F9E1220920E1220771
:1812230020D1CD9416CD3F10CDB916CD9E152B220F20215218C3851845
:18123B00CD2A10221320215A18C38518CD2A10C35012210100221120AB
:181253002A0720221520EB221720019A002A0F20EB6068393E097E235F
:18126B06B6CA89127E2BBAC268127EBBC26812EB21000039444D210A3B
:181283000019CD8516F92A1720EBCD3F10CD4610DA4015220B20D5EB12
:18129B002A0F207CB5CA4115CD3110CAB312D1CD94162A0B20C39912E9
:1812B3005E23562A1120E57CAA7A19FAC512ACFAE912EB2A0F29732307
:1812CB00722A1320F1B7F2D512EBCD0415D1DAEB122A15202207202A70
:1812E3001720EBCD3F19E1D1CD9416CD3F10210000C3FA12CD2A107CFD
:1812FB00B5C27611CDD915D26F11C3D5102A0D20F9E1220720D1D1D537
:18131300CDF115C32213CD4610DA6013C33413D5CD4610DA40151A4FED
:18132B009712D1CDE315791B12D5EB2A0720E52112132207202100001F
:1813430039220D20D53E3ACD7615CDB518CD2A10CD751AD1EB732372A4
:18135B00F1220720D1F1CD17102C03C31213CD3F101AFE0DCA7D13CD1B
:181373000E15CD17102C03C37213CD3F10216418C38518CDAF13D86FE5
:18138B00C9CDAF13C86FC9CDAF13C8D86FC9CDAF136FC8D86CC9CDAF3C
:1813A30013C06FC9CDAF13D96FC9E1C979E1C1E5C54FCDC413EBE3CD93
:1813BB000415D12100003E01C9CD17102D06210006C3F613CD17102BD4
:1813D30000CD0014CD17102B15E5CD0014EBE37CAA7A19D1FAD713AC3F
:1813EB06F2D713C3B310CD17102D92E5CD0014CDEF14C3E613CD641444
:18140300CD17102A2DE5CD64140600CDEC14E3CDEC14EBE37CB7CA22EC
:18141B00147AB2EBC2B4107D210000B7CA561419DAB4103DC22A14C3C8
:181433005614CD17102F4EE5CD64140600CDEC14E3CDEC14EBE3EB7AE6
:18144B00B3CAB410C5CDCF146069C1D17CB7FAB31078B7FCEF14C30394
:1814630014210A18C38518CD4610DA75147E23666FC9CD891078B7C0A0
:18147B00CD17102809CD2A10CD17102901C9C34015CD7B147CB7FAB3F2
:1814930010B5CAB310D5E52A1920110520CD3110DAA914210F105E2336
:1814AB0056221920E1EBC5CDCF14C1D123C9CD7B141BCDEC1413C92A6F
:1814C3001B20D5EB2A2A21CDE514D1C9E56C2600CDDA14417DE1670EFB
:1814DB00FF0CCDE514D2DC1419C97D936F7C9A67C97CB7F67CB5C87C27
:1814F390F52F677D2F6F23F1ACF2B31078EE8047C97CAAF20A15EBCDE1
```

Abb. 2.4.1.2-2 Programm im Intel-HEX-Format

```
:18150B003110C9CD4610DA4015E5CD17103D6ACD2A16444DE1712376CF
:18152300C9C34015CD17103B04F1C37611CD17100D04F1C36611C9CD9B
:18153B003710FE0DC8D511C71097CDE315D11AF597122A0720E57E2305
:18155300B6D1CAD5107EB7FA0813CD65161BF1123E3FCD211097CDE3D8
:18156B0015C3D510D511CE10C34415CD2110CDB518CD2C17FE01CAA1B9
:1815830015CD2110FE0ACA7C15B7CA7C15FE1BCAB1151213FE0DC87BAC
:18159B00CDC518C27C157BCDCC18CAB1151B3E08CD2110C37C15CD1FE0
:1815B300103E0BC376157CB7FAB310113021E52A1B202BCD3110E1D8EB
:1815CB001A9547131A9CDAD8151BB0C913131AFE0DC2D81513C3C11548
:1815E300471A13B8C8CD2110FE0DC2E415C9CD1710220F3E22CDE31525
:1815FB00FE0DE1CA6611232323E9CD171627053E27C3F815CD17165FB1
:181613000C3E8DCD2110CD2110E1C30116C90600CDEC14F22C16062D2E
:18162B000DD5110A00D50DC5CDCF1478B1CA4316E32DE56669C333163D
:18164300C10D79B7FA52163E20CD2110C3441678B7C421105D7BFE0AB2
:18165B00D1C8C630CD2110C358161A6F131A67130E04CD21163E20CD48
:18167300211097CDE315C9CD3110C81A921303C37A167892C28D1679C6
:18168B0093C81B2B1A77C38516C1E1220F207CB5CAAE16E1221120E1F0
:1816A300221320E1221520E1221720C5C9212720CDEF14C139D26F1552
:1816BB002A0F207CB5CAD6162A1720E52A1520E52A1320E52A1120E5CB
:1816D3002A0F20E5C5C93206291603CD1F1015C2DE1697114617CDE346
:1816EB0015210F1022192021302122 1B20218A23222A21218C23222C0F
:181703002121CC23222E21C3D510C21317F1C1C9F1F54F79FE0DCA2278
:18171B0017CD0910F1C1C90E0DCD09160E0AC31C17CD6616E67FFE62E7
:18173300C240173A06202F320620C32C17FE03C0C3D51052444B2050DE
:18174B00524F4D50542042415349432056332E3220334B6D6A4C4953CC
:181763005400901152554E0060114E4557005111425945000419454E37
:18177B004400D3184E4558540090124C4554007213494600F712474FAE
:18179300544F007F11474F53554209EE1152455455524E0010125245F3
:1817AB004D00F112464F52002B12494E5655540012135052494E540070
:1817C300A81153544F50005A1143414C4C0008194F5554434841520051
:1817DB001A1A4F55540015194F240066194924007319564F4B4500ACD6
:1817F30019544142003B194259544500CE19574F524400D819006C13D3
:18180B00524E44008C1441425300B91453495A4500C2145045454B0068
:18182300A519494E4348415200241A484558002B1A494E004C192700AF
:18183B00021A544F5006101A4C454E00151A4353545300FA19006A1480
:18185300544F003B1200401553544550004712904D123E3D0086132 30D
:18186B00008C133E0092133D00A1133C3D0099133C00A71366AD1321F6
:181883005F17CD3710D51A13FE2ECAA71823BECA09183E001BBECAAE31
:18189B001823BEC29C182323D1C385183E0023BEC2A918237E23666F14
:1818B300F1E9E52A2C21545DE1C9E52A2A21545DE1C9E52A2E21BDE1DB
:1818CB00C9E52A2C21BDE1C9CD2A10EB218A23EBCD3110DA76157CB72E
:1818E300FA70157E2F7746B8C27015222E217DD6846F7CDE0067222C3F
:1818FB00212B2B222A21C3D510FFC3D510CD2A16D5011119C5E9D1CD4F
:181913003F10CD7B14E5CD17103D1ACD2A10453ED3320220E17D32039D
:18192B00203EC932042078CD0220CD3F10C34015CD7B147CB5CC3F10E4
:181943002B3E20CD2110C33E19CD7B14E53EDB320220E17D3203203E4C
:18195B00C9320420CD022026066FC9CD2A10D5EBAFCDE315D1CD3F10E0
:18197300CD2A10D5EB2A1B20EBCD3110DA7015CDB518CD7C15444DEB64
:18198B002BCDB518D5CD7A16AF02D123CDE514EB2100207323 72D1CD10
:1819A3003F10CD7B146E2600C9CD2A10D5EB2A1B20EBCD3110DA7015A0
:1819BB00D1E5CD17102C09CD2A107DE177CD3F10C34015CD7B147DCD7F
:1819D300E619CD3F10CD7B147CCDE6197DCDE619CD3F10F50F0F0F0FA7
:1819EB00CDEF19F1E60FC69027CE4027C32110CD6C102F6F2600C91AF3
:181A0300136F2600CD17102701C9C340152A1B2023C92A00202BC9CDCA
:181A1B002A107DCD2110CD3F10CD2C1726006FC9C5210000CD17102872
:181A33001D1AFE0DCA4015CD561A2929292906004F0913CD17102903C8
:181A4B00C3541AC3341AC34015C1C9FE30FA4015FE39FA6D1ACA6D1A19
:181A6300FE41FA4015FE47F24015D630FE0AF8D607C9CD0C10C8CD0621
:081A7B0010FE03C0C3D51000EA
:0000000000
```

zu Abb. 2.4.1.2-2

```
                        .PABS
                        .PHEX
                        ;*************************************************
                        ***
                        ;  RDK  B A S I C  INTERPRETER     V3.2 780524
                        ;*************************************************
                        ***
0100                    .LOC 100H
0100    C3 0FE3         JMP BEGINN
0FE3                    .LOC 0FE3H
0FE3                    BEGINN:
0FE3    31 29F1         LXI SP,STACK
0FE6    3EC9            MVI A,0C9H
0FE8    32 1000         STA 1000H
0FEB    CD 1000         CALL 1000H
0FEE                    ANF:
0FEE    3B              DCX SP
0FEF    3B              DCX SP
0FF0    D1              POP D
0FF1    21 0012         LXI H,HAUPTP-ANF
0FF4    19              DAD D
0FF5    11 1000         LXI D,1000H
0FF8    01 0A9F         LXI B,ENDE-BEGINN
0FFB    EDB0            LDIR
0FFD    C3 1000         JMP 1000H
                        ;HAUPTPROGRAMM
1000                    HAUPTP:
1000    C3 100F         JMP START
1003    C3 1005         JMP RSTART
1006    C3 F003         CI:JMP 0F003H
1009    C3 F009         ECHO:JMP 0F009H
100C    C3 F012         CSTS:JMP 0F012H
100F                    START:
100F    31 29F1         LXI SP,STACK
1012    3EFF            MVI A,0FFH
1014    C3 1609         JMP INIT
```

Abb. 2.4.1.2-3 Anfang des Programmlistings

Z80-Befehl, doch der INTERPRETER selbst ist in der Sprache des 8080 geschrieben. Will der Anwender von der Verschiebemethode Gebrauch machen, so muß er diesen Anfang einfach vor den INTERPRETER schreiben und im ROM mit ablegen.
Das Programm kann aber auch direkt auf die Adresse 1ØØØH im ROM abgelegt werden.

Der BASIC INTERPRETER beginnt mit einem Sprung auf START mit der Adresse 1ØØØH. Dort wird er aufgerufen, wenn er nach dem Einschalten des Computers zum ersten Mal aufgerufen wird. Bei weiteren Aufrufen kann er auf der Adresse 1ØØ3H gestartet werden, die den Namen RSTART hat.

Dabei wird ein evtl. zuvor abgelegtes Programm nicht gelöscht und bleibt dem Benutzer erhalten. Auf der Adresse 1ØØ6H steht ein Sprung zu einer Routine mit dem Namen CI, die der Anwender selbst eintragen muß. Diese Routine muß im Register A ein Zeichen von der Konsole im 7-Bit-Code (ASCII) ablegen. Andere Register dürfen nicht verändert werden. Das Paritäts-Bit muß auf 0 gesetzt sein (Bit 7). Auf der Adresse 1ØØ9H steht ein Sprung zur Routine CO, die die Aufgabe hat, ein Zeichen auf die Konsole auszugeben. Das Zeichen wird dabei im Register C geliefert und muß nach Aufruf der Routine auch im Register A stehen. Andere Register dürfen nicht verändert werden. Der Sprung auf die Adresse 1ØØCH führt zu einem Unterprogramm mit dem

Namen CSTS. Es hat die Aufgabe, zu überprüfen, ob an der Konsole ein Zeichen eingegeben wurde. Ist dies der Fall, so muß im Register A der Wert ØFFH stehen, sonst der Wert Ø. Andere Register dürfen nicht verändert werden.

2.4.2 12 K BASIC

Hier soll noch ein größeres BASIC für Mikrocomputer besprochen werden, das den gesamten BASIC-Sprachumfang erfüllt und darüber hinaus noch zusätzliche Befehle enthält.

Tabelle 2.4.2-1 zeigt den gesamten Befehlssatz des 12K TDL BASIC von Technical Design Labs. BASIC Systeme mit Besonderheiten sind z.B. die Funktionen PEEK und POKE, wie auch die Befehle INP und OUT. Dabei können PEEK und POKE auch auf BASIC Systemen anderer Mikrocomputer für andere Prozessoren (z.B. 6800) vorkommen, doch die Befehle INP und OUT sind sehr spezifisch für 8080 und Z80. Weitere interessante Befehle sind hier FNEND und FNRETURN. Damit ist es möglich, Rekursive Funktionen zu definieren.

Beispiel:
```
100 DEF FNFAC (I)
200 IF I=0 THEN FNRETURN 1
300 FNEND FNFAC(I-1)*I
```

Die klassische Recursive Funktion stellt die Fakultätsbildung dar. Es gilt FAC(0)=1 und FAC(I)=FAC(I-1)*I. Im Beispiel ist gezeigt, wie es möglich ist, die Funktion FAKULTÄT zu definieren.

Tabelle 2.4.2-1

ABS	Absolutwert bilden
ALOAD	Laden eines ASCII-Programmes
ALOADC	Laden eines ASCII-Programmes mit gesteuertem-Leser
AMERGE	Hinzuladen eines ASCII-Programms
AMERGEC	Hinzuladen mit gesteuertem-Leser
AND	Logisch UND-Verknüpfung
ASAVE	Ausgabe eines Programms in ASCII
ASC	Zeichen in numerischen Wert konvertieren
ATN	arctan-Funktion
AUTO	Automatische Zeilennummerierung
CALL	Aufruf eines Maschinenunterprogramms
CHR$	Numerischen Wert in ASCII-Zeichen wandeln
CLEAR	Alle Variablen löschen und Speicher für Strings reservieren
CONT	Programmausführung nach Unterbrechung fortsetzen
COPY	BASIC-Programmteile verschieben
COS	cos-Funktion
DATA	Definition von Konstanten
DEF	Benutzerfunktion (auch mehrzeilig rekursiv)
DELETE	Löschen
DIM	Dimensionierung von Feldern
EDIT	Aufruf des Zeilen-Editors
ELSE	Konstruktion IF . . THEN . . ELSE . .

END	Programmende
EXCHANGE	Austausch von Variablenwerten
EXP	Exponentialfunktion
FN	Benutzerfunktion
FNEND	Ende einer mehrzeiligen Funktion
FNRETURN	Wertübergabe vorzeitig in Funktion
FOR	Beginn einer Schleife
FRE	Freier Speicherraum
GOSUB	Unterprogrammaufruf
GOTO	Sprunganweisung
IF	Bedingte Anweisung
INP	Eingabe von E/A Port
INPUT	Eingabe von der Tastatur
INSTR	Teilstring in String suchen
INT	Integer-Funktion
KILL	Speicherreservierung aufheben
LEFT$	Linker Teil eines Strings
LEN	Länge eines Strings
LET	Zuweisung
LINE INPUT	Eingabe formatfreier Texte
LIST	Programm ausdrucken auf Konsole
LLIST	Programm ausdrucken auf Drucker
LLVAR	Variablen ausdrucken auf Drucker
LNULL	NUL-Zeichen auf Drucker ausgeben
LOAD	Programm vom Leser laden im Internformat
LOADGO	BASIC-Programm lädt ein anderes Programm UND übergibt die Steuerung
LOG	Natürlicher Logarithmus
LPOS	Position des Druckkopfes
LPRINT	Ausgabe auf Drucker
LVAR	Variable auf Konsole ausgeben
LWIDTH	Druckerbreite einstellen
MID$	Mittelteil eines Strings
NEW	Programm und Variable löschen
NEXT	Rückkehr zum Schleifenbeginn
NOT	logisches NICHT
NULL	NUL-Zeichen auf die Konsole geben
ON	Mehrfach verzweigter Sprung
OR	logisches ODER
OUT	Ausgabe über E/A Port
PEEK	Direkter Zugriff zum Speicher
POKE	Direktes Einschreiben in Speicher
POS	Schreibposition der Konsole
PRECISION	Einstellen der Dezimalstellen (max. 12)
PRINT	Ausgabe über Konsole
PRINT USING	Formatierte Ausgabe
RANDOMIZE	Zufallsfunktion einstellen
READ	Dateien von DATA-Zeile lesen
REM	Bemerkung

RENUMBER	Zeilennummern ändern (auch Bereichsweise)
RESTORE	DATA-Zeiger rücksetzen
RETURN	Rückkehr von Unterprogramm
RIGHT$	Rechter Teil eines Strings
RND	Zufallsfunktion
RUN	Programmstart
SAVE	Retten eines Programms im Internformat
SGN	Vorzeichen einer Variablen
SIN	sin-Funktion
SPC	Leerzeichen in der PRINT-Anweisung
SQR	Wurzel
STEP	Schrittweite in Schleifen
STOP	Programmausführung beenden
STRS	Numerischen Wert in ASCII-String wandeln
SWITCH	Konsolenzuweisung ändern
TAB	Position des Druckkopfes in der PRINT Anweisung einstellen
TAN	tan-Funktion
THEN	für IF THEN
TO	für Schleifenanweisung
TRACE	Zeilennummern der ausgeführten Befehle ausdrucken
USR	Benutzermaschinenprogramm-Funktion
VAL	String in numerischen Wert konvertieren
WAIT	Status-Port abfragen für E/A
WIDTH	Schreibbreite der Konsole einstellen
&	Konstante wird sedezimal interpretiert
?	Entspricht PRINT-Kurzschreibweise
^	Exponential-Operator
-	Minus
*	Mal
/	Geteilt durch
+	Plus
<	Kleiner als
>	Größer als
<>	Ungleich
=	Gleich
CTRL U	Zeile löschen
CTRL C	Programm unterbrechen
CTRL X	Rückkehr zum Monitor
CTRL O	Konsolausdruck unterbinden
CTRL R	Weitere Eingabezeichen
CTRL T	Ausdruck der gerade ausgeführten Zeilennummer
CTRL S	Zeitweise Unterbrechung
CTRL Q	Wiederstart des Programms
RUBOUT	Vorhergehendes Zeichen löschen
,	Druckkopf zur nächsten TAB-Position
;	Druckkopf bleibt an der selben Stelle
:	Trennung von Mehrfachbefehlen

Das hier beschriebene System BASIC ist ähnlich wie das vereinfachte RDK BASIC über eine Sprungtabelle angepaßt und wird im Relocating Format geliefert, so daß es auf eine beliebige Speicheradresse geladen werden kann. Nur eine Forderung schränkt die Speicherwahl etwas ein. Der INTERPRETER benötigt einen RAM-Buffer von 1ØØH bis 2FFH. BASIC ist unter dem Namen XITAN BASIC in einer erweiterten Form auch für das CP/M Disketten-Betriebssystem lieferbar. Es benötigt dann einen Speicher von 19 KByte für den INTERPRETER. Eine Funktion, die dann hinzukommt, ist z.B. OPEN zum Öffnen von Dateien auf der Disk-Platte oder einer anderen Struktur bei den Ein- und Ausgabeanweisungen. So wird ein bequemer Verkehr mit Disk- oder anderen Peripherie-Geräten möglich. Ferner kommen interessante Funktionen für den Programmierablauf hinzu: FIND und REPLACE. Mit FIND ist es möglich, nach einem beliebigen Programmtext innerhalb des gesamten BASIC Programms zu suchen. Mit REPLACE kann dieser gegen einen anderen ersetzt werden. Weitere Funktionen sind z.B. DATE und TIME, mit denen Zeitangaben abgefragt werden können, falls die dazugehörige REAL TIME CLOCK im System als Hardware vorhanden ist.

2.5 Disassembler

Oft besteht der Wunsch, ein Programm wieder in Mnemonics vom Assembler zurück zu übersetzen, das im Maschinencode vorhanden ist. Dies kann z.B. bei fremden Programmen erforderlich sein, um Systemanpassungen vorzunehmen.

Ein Disassembler hat eine komplizierte Aufgabe. Er muß versuchen, sich in einen Programmierer „hineinzudenken", um den Assembler-Code zu regenerieren. Daß dies nicht immer gelingt, ist selbstverständlich. Ein so erzeugtes Listing muß meist neu überarbeitet werden.

Die Probleme ergeben sich mit der Sprungmarke und den Datenbereichen, da Datenbereiche vom Objektcode her gesehen nicht vom Programm unterschieden werden können. Um Sprungmarken zu erkennen, ist es nötig, ähnlich wie beim Assembliervorgang, mit zwei Durchläufen (Pass) zu arbeiten. Dabei wird beim ersten Pass eine Symboltabelle aufgebaut, die mit selbst erfundenen Marken besetzt ist. Dazu wird der Objektcode geladen und so gut wie möglich versucht, alle Sprungbefehle zu erkennen. Dabei besteht natürlich auch die Gefahr, Datenbereiche mit zu interpretieren. Beim zweiten Pass wird zu jedem erkannten Objektcode eine Marke hinzugefügt, falls diese in der Symboltabelle vorhanden ist, und der Objektcode in einen Assemblerbefehl disassembliert.

Um dem Problem mit den Datenbereichen zu begegnen, ist es auch möglich, einen Mehrpass-Disassembler zu konstruieren. Dieser versucht dann, das Programm teilweise auszuführen, insbesondere alle Kombinationen möglicher Sprünge auszuführen. Dann werden alle angesprungenen Gebiete vermerkt und als Programmgebiet gekennzeichnet. Doch auch ein solcher Disassembler ist nicht unfehlbar, denn Befehle, wie PCHL, bei denen die Sprungadresse erst errechnet wird, bereiten ihm Schwierigkeiten.

2.5.1 Universal Disassembler in BASIC

An dieser Stelle soll ein Disassembler beschrieben werden, der sich für mehrere Prozessor-Sprachen eignet. Der Disassembler ist in BASIC geschrieben, so daß er relativ leicht auch auf anderen Prozessoren laufen kann. Dabei wurde das 19K XITAN BASIC verwendet. Ein Disc BASIC wird empfohlen, ist jedoch nicht unbedingt Voraus-

```
10 REM UNIVERSAL DISASSEMBLER FUER VERSCHIEDENE PROZESSOREN
20 REM BY ROLF-D. KLEIN V790610 1.0
30 '
40 '
50 CLEAR 3000,2
60 DV=0          'DEVICE=CONSOLE
100 DIM M$(256),A(256),R(256),S(1000)
110 INPUT "PROZESSOR TYP=?";P$
120 OPEN #20,"I",P$+".DIS"
130 I=0 'LESE ZAEHLER
140 ON EOF GOTO 280
150 A$=""
160 B$=BYTE$(#20)
170 IF B$="" "THEN 220 'TERMINATOR
180 IF (B$=CHR$(13))OR(B$=CHR$(10))THEN 140
190 IF B$="-" THEN B$=" "
200 A$=A$+B$
210 GOTO 160
220 M$(I)=A$
230 A(I)=VAL(BYTE$(#20)) 'ANZAHL BYTES
240 IF BYTE$(#20)<>" " THEN PRINT#DV,"EINGABEFEHLER"
250 R(I)=VAL(BYTE$(#20)) 'RELATIV INDEX
260 I=I+1 'INDEX POINTER
270 IF A$<>"END" THEN 150
280 CLOSE #20
290 '
300 INPUT "QUELLE DISK=1 MEM=0";Q
310 IFQ=0THENINPUT"ANFADR,ENDADR";A,E:E=E+1:GOTO 350
320 INPUT "QUELLDATEI";Q$
330 OPEN #20,"I",Q$
340 '
350 INPUT "ZIEL DISK=1 SONST=0";Z
360 IF Z=0 THEN 390
370 INPUT "ZIELDATEI";Z$
380 OPEN #21,"O",Z$
385 OPTION #21,"O",0
390 '
395 INPUT "DRUCKER=2,PUNCH=4,CONSOLE=0";DV
400 PRINT#DV,"PASS 1"
410 A1=A          'ZWISCHENSPEICHER ANFANGSADRESSE
420 S1=0          'SYMBTAB POINTER
430 INPUT "PROGRAMMANFANGSADRESSE ";P1
440 P2=P1         'RETTEN PC
450 '
500 GOSUB 10000 'HOLEN EINES BYTES IN B
510 IF B=999 THEN 1000   'ENDE DER FILE
520 IF R(B)=1 THEN 600
530 IF R(B)=2 THEN 700
540 IF R(B)=3 THEN 800
550 P1=P1+A(B)   'PC INCREMENTIEREN
560 IF A(B)=1 THEN GOTO 500      'NAECHSTES BYTE
570 FOR I=1 TO A(B)-1
580 GOSUB 10000:NEXT I   'REST WEGSCHMEISSEN
590 GOTO 500             'WEITER LESEN
600 'RELATIVE ADRESSE
610 GOSUB 10000 'BYTE HOLEN
620 P1=P1+2      'NEUER PC
630 IF B<&80 THEN B=B+P1:GOTO 645        'BERECHNEN ADRESSE
640 IF B>=&80 THEN B=(B-256)+P1 'ZWEI FAELLE
645 IF B>&FFFF THEN 670
650 PRINT#DV,"RELADR:",HEX$(B)
660 GOSUB 11000 'EINTRAGEN IN SYMBOLTABELLE
670 GOTO 500     'WEITER
700 '
710 'ABSOLUTE ADRESSE REVERSE
```

Abb. 2.5.1-1 Universal Disassembler in BASIC

```
715 P1=P1+3
720 GOSUB 10000 'BYTE HOLEN
730 C=B 'UMSPEICHERN
740 GOSUB 10000 'HIGH ADR
750 B=C+256*B    'ABSOL ADR
755 IF B>&FFFF THEN 780
760 PRINT#DV,"ABSADR",HEX$(B)
770 GOSUB 11000 'SYMBOLTABELLE
780 GOTO 500     'WEITER
800 '
810 'ABSOLUTE ADRESSE FORWARD                    zu Abb. 2.5.1-1
815 P1=P1+3
820 GOSUB 10000 'BYTE HOLEN
830 C=B 'UMSPEICHERN
840 GOSUB 10000 'LOW ADR
850 B=B+256*C
855 IF B>&FFFF THEN 880 'ENDE FILE
860 PRINT#DV,"ABSADR",HEX$(B)
870 GOSUB 11000 'SYMBOLTABELLE
880 GOTO 500     'WEITER
890 '
1000 '
1010 'PASS 2
1020 PRINT#DV,"PASS 2"
1030 P1=P2       'ALTER PC
1040 S2=0        'SYMBOLTABELLE
1050 IF Q=0 THEN 1080
1060 CLOSE #20
1070 OPEN #20,"I",Q$
1080 '
1090 A=A1        'ADRESSE AUF ANFANG
1100 F1=0        'SORTIERFLAG
1110 FOR J=0 TO S1-1     'SORTIEREN
1120 IF S(J)>S(J+1) THEN EXCHANGE S(J),S(J+1):F1=1
1130 NEXT J
1140 IF F1=1 THEN 1100   'NOCH NICHT SORTIERT
1150 '
1160 PRINT#DV,"OK SYMBOLTABELLE SORTIERT "
1170 FOR J=0 TO S1
1180 PRINT#DV,HEX$(S(J)),
1190 NEXT J
1192 PRINT#DV
1195 '
1200 'NUN PASS 2 DURCHFUEHREN
1210 B1$="":B2$="":B3$="":B4$=""          'VORBELEGEN
1220 M$="":L$="":M1$=""
1225 AD$=HEX$(P1)        'ADRESSE
1230 GOSUB 10000         'BYTE HOLEN
1235 IF B=999 THEN 3000 'ENDE
1240 B1$=RIGHT$(HEX$(B),2)
1250 M$=M$(B)    'MNEMONIC
1260 GOSUB 13000         'SYMBOLTABELLE DURCHSUCHEN
1270 IF R(B)=1 THEN 1400         'RELADR
1280 IF R(B)=2 THEN 1500
1290 IF R(B)=3 THEN 1600
1295 '
1300 'NORMALER OPCODE
1305 P1=P1+A(B)
1306 N=B         'RETTEN B
1310 IF A(N)=1 THEN 2000
1320 GOSUB 10000         'NEUEN WERT
1330 B2$=RIGHT$(HEX$(B),2):L$="0"+HEX$(B)+"H":C=B
1340 IF A(N)=2 THEN 2000
1345 GOSUB 10000
1350 B3$=RIGHT$(HEX$(B),2):L$="0"+HEX$((C+256*B)+"H"      'LOW HIGH
1360 IF A(N)=3 THEN 2000
```

```
1370 PRINT#DV,"FEHLER":GOTO 2000
1375 '
1400 'RELATIV ADR
1410 P1=P1+2     'NEUER PC
1420 GOSUB 10000          'WERT HOLEN
1430 B2$=RIGHT$(HEX$(B),2)
1440 IF B<&80 THEN B=B+P1:GOTO 1460
1450 IF B>=&80 THEN B=(B-256)+P1
1455 IF B>&FFFF THEN 2000
1460 L$="M"+HEX$(B)
1470 GOTO 2000
1500 'ABS ADR MIT LOW HIGH
1510 P1=P1+3     'NEUER PC
1520 GOSUB 10000          'LOW ORDER
1530 C=B         'RETTEN
1540 GOSUB 10000
1545 B2$=RIGHT$(HEX$(C),2):B3$=RIGHT$(HEX$(B),2)
1550 B=C+256*B   'ADRESSE BERECHNEN
1555 IF B>&FFFF THEN 2000
1560 L$="M"+HEX$(B)
1570 GOTO 2000
1575 '
1600 'ABS ADR MIT FORWARD
1610 P1=P1+3
1620 GOSUB 10000
1630 C=B
1640 GOSUB 10000
1645 B2$=RIGHT$(HEX$8C),2):B3$=RIGHT$(HEX$(B),2)
1650 B=B+256*C
1660 IF B>&FFFF THEN 2000
1670 L$="M"+HEX$(B)
1680 GOTO 2000
2000 '
2010 GOSUB 12000          'AUSGABE
2020 GOTO 1200
3000 IF Z=1 THEN PRINT#21,".END"
3010 CLOSE                'ALLE FILES SCHLIESSEN
3020 END                  'ENDE
10000 'GET BYTE
10010 IF Q=1 THEN 10060
10020 B=PEEK(A)
10030 A=A+1      'ADRESSPOINTER
10040 IF A>E THEN B=999 'ENDE KRITERIUM
10050 RETURN
10060 B=BYTE(#20)         'VON FILE LESEN
10070 ON EOF GOTO 10090
10080 RETURN
10090 B=999
10100 RETURN
11000 '
11010 'SYMBOLTABELLEN EINTRAG
11020 S(S1)=B    'EINTRAGEN
11030 S1=S1+1    'AUF NAECHSTE FREIE ADR
11040 RETURN
12000 '
12010 'AUSDRUCKEN
12020 PRINT#DV,USING 12030;AD$,B1$,B2$,B3$,B4$,M1$,M$,L$
12030 ' 'LLL  'L 'L 'L 'L     'LLLLL 'LLLLLLLLL 'LLLLLL
12040 IF Z=0 THEN RETURN
12050 PRINT#21,M1$;:OUTBYTE#21,9:PRINT#21,M$;:OUTBYTE#21,9:PRINT#21,L$
12070 RETURN
13000 'SUCHEN
13010 IF S2>-S1 THEN RETURN
13020 IF S(S2)=P1 THEN M1$="M"+HEX$(P1)+":":RETURN
13030 IF S(S2)<P1 THEN S2=S2+1 :GOTO 13010
13040 RETURN
```

zu Abb. 2.5.1-1

setzung für die Funktionsfähigkeit dieses Assemblers. Es handelt sich um einen Zwei-Pass-Disassembler, der also auch Marken generiert.

Abb. 2.5.1-1 zeigt das Listing des Disassemblers. Das Programm beginnt mit dem Befehl Clear 3000,2, der bedeutet, daß 3000 Byte für Strings reserviert werden. Die Zahl 2 bedeutet, daß maximal 2 Datenfiles gleichzeitig eröffnet werden können. Die Zuweisung auf Zeile 60 an DV bewirkt, daß alle Ausgaben auf die Konsole geleitet werden, da DV als logische Gerätenummer weiter verwendet wird. Bei M$(256) werden 256 Plätze für das String-array M$ reserviert, das zur Aufnahme der Mnemonics dient. A ( ) beinhaltet die Anzahl Byte, die ein Befehl benötigt, R ( ) gibt an, ob es sich um eine Adresse handelt oder um eine Konstante. Falls es eine Adresse ist, bedeutet 1 relative Adresse mit einem Byte, 2 absolute Adresse mit LOW HIGH Folge (8080, Z80) und 3 Adresse mit HIGH LOW Folge (6800), beidemal mit 2 Byte für die Adresse.

In 110 wird der Typ des Prozessors angefragt. Dieser wird in eine String-Variable mit dem Namen P$ eingelesen. Anschließend wird in 120 eine Datei mit dem Namen des Prozessors und der Extension .DIS eröffnet, die die spezifischen Daten für den Prozessor und Mnemonic enthält.

Falls der Anwender kein Disk BASIC zur Verfügung hat, so kann er die Daten auch mit Hilfe einer DATA-Anweisung in die ARRAYS M$, A ( ) und R ( ) einlesen. Die Zeilen 130 bis 280 sind relativ BASIC spezifisch und haben nur die Aufgabe, die Mnemonics in das Array M$ und die Längenangaben in A ( ) und die Datentypangabe in R ( ) einzulesen.

Dabei wurde das Format auf der Disk so gewählt, daß es leicht mit einem EDITOR einzugeben und weitgehend formatfrei ist. Dabei werden in Mnemonics vorkommende Blanks durch „-" ersetzt und Mnemonics werden durch Blanks von Daten getrennt.

Abb. 2.5.1-2 zeigt ein Beispiel für einen 8080-Datensatz, der um Z80 Mnemonics erweitert wurde. In Zeile 300 bis 340 wird bestimmt, ob aus dem Speicher oder von der Disk disassembliert werden soll. Falls vom Speicher gelesen wird, so wird eine Anfangs- und eine Endadresse erfragt, falls von der Disk, so muß nur ein Dateiname angegeben werden. Dieser wird in der Stringvariablen Q$ gespeichert. In den Zeilen 30 bis 385 wird festgelegt, ob die Ausgabe der Assemblerquelle auf Disk erfolgen soll, oder keine Ausgabe erfolgt. In Zeile 395 wird angefragt, ob die Ausgabe eines Listings (im Code dem Assembler ähnlich) auf den Drucker, auf Punch oder auf Konsole erfolgen soll. Mit Zeile 400 beginnt der erste Durchlauf des Disassemblers, der der Aufnahme der Symboltabelle dient. In Zeile 430 wird die echte Adresse für den Programmanfang des zu disassemblierenden Programms erfragt. Diese kann von der physikalischen Anfangsadresse differieren, falls aus dem Speicher oder von der Disk disassembliert wird. Die Subroutine 10000 hat die Aufgabe, ein Byte von der Disk oder vom Speicher zu holen. Falls kein Byte mehr verfügbar ist, wird der Wert 999 nach B übergeben, der in dem normalen Bereich von 0 bis 255 der Bytes nicht vorkommt. Mit P1 wird der Programmzähler bezeichnet. Nach dem Einlesen des ersten Byte wird überprüft, ob ein Symboleintrag nötig ist oder nicht. Ist R(B)=0, so ist kein Eintrag nötig. Der Programmzähler wird um A(B) erhöht, wobei A(B) die Länge des Befehls angibt. Die Datenbytes eines Befehls werden in den Zeilen 570 bis 590 ignoriert durch n-maligen Aufruf des Unterprogramms 10000. Handelt es sich aber um einen Befehl mit Adresse, so wird je nach Zugehörigkeit nach 600, 700 oder 800 gesprungen, wobei die Zeile 600 relative Adressenkonstanten enthält. Dazu wird dort zunächst das nächste Byte geholt, das den relativen Adreßoffset darstellt und in B steht. Der Programmzähler wird um 2

```
A>TYPE 8080.DIS

NOP 1 0          DAD-SP 1 0       MOV-M,D 1 0      XRA-E 1 0        RST-2 1 0
LXI-B, 3 0       LDA 3 0          MOV-M,E 1 0      XRA-H 1 0        RC 1 0
STAX-B 1 0       DCX-SP 1 0       MOV-M,H 1 0      XRA-L 1 0        EXX 1 0
INX-B 1 0        INR-A 1 0        MOV-M,L 1 0      XRA-M 1 0        JC 3 2
INR-B 1 0        DCR-A 1 0        HLT 1 0          XRA-A 1 0        IN 2 0
DCR-B 1 0        MVI-A, 2 0       MOV-M,A 1 0      ORA-B 1 0        CC 3 2
MVI-B, 2 0       CMC 1 0          MOV-A,B 1 0      ORA-C 1 0        DD 2 0
RLC 1 0          MOV-B,B 1 0      MOV-A,C 1 0      ORA-D 1 0        SBI 2 0
EXAFAF 1 0       MOV-B,C 1 0      MOV-A,D 1 0      ORA-E 1 0        RST-3 1 0
DAD-B 1 0        MOV-B,D 1 0      MOV-A,E 1 0      ORA-H 1 0        RPO 1 0
LDAX-B 1 0       MOV-B,E 1 0      MOV-A,H 1 0      ORA-L 1 0        POP-H 1 0
DCX-B 1 0        MOV-B,H 1 0      MOV-A,L 1 0      ORA-M 1 0        JPO 3 2
INR-C 1 0        MOV-B,L 1 0      MOV-A,M 1 0      ORA-A 1 0        XTHL 1 0
DCR-C 1 0        MOV-B,M 1 0      MOV-A,A 1 0      CMP-B 1 0        CPO 3 2
MVI-C, 2 0       MOV-B,A 1 0      ADD-B 1 0        CMP-C 1 0        PUSH-H 1 0
RRC 1 0          MOV-C,B 1 0      ADD-C 1 0        CMP-D 1 0        ANI 2 0
DJNZ 2 1         MOV-C,C 1 0      ADD-D 1 0        CMP-E 1 0        RST-4 1 0
LXI-D 3 0        MOV-C,D 1 0      ADD-E 1 0        CMP-H 1 0        RPE 1 0
STAX-D 1 0       MOV-C,E 1 0      ADD-H 1 0        CMP-L 1 0        PCHL 1 0
INX-D 1 0        MOV-C,H 1 0      ADD-L 1 0        CMP-M 1 0        JPE 3 2
INR-D 1 0        MOV-C,L 1 0      ADD-M 1 0        CMP-A 1 0        XCHG 1 0
DCR-D 1 0        MOV-C,M 1 0      ADD-A 1 0        RNZ 1 0          CPE 3 2
MVI-D, 2 0       MOV-C,A 1 0      ADC-B 1 0        POP-B 1 0        ED 2 0
RAL 1 0          MOV-D,B 1 0      ADC-C 1 0        JNZ 3 2          XRI 2 0
JMPR 2 1         MOV-D,C 1 0      ADC-D 1 0        JMP 3 2          RST-5 1 0
DAD-D 1 0        MOV-D,D 1 0      ADC-E 1 0        CNZ 3 2          RP 1 0
LDAX-D 1 0       MOV-D,E 1 0      ADC-H 1 0        PUSH-B 1 0       POP-PSW 1 0
DCX-D 1 0        MOV-D,H 1 0      ADC-L 1 0        ADI 2 0          JP 3 2
INR-E 1 0        MOV-D,L 1 0      ADC-M 1 0        RST-0 1 0        DI 1 0
DCR-E 1 0        MOV-D,M 1 0      ADC-A 1 0        RZ 1 0           CP 3 2
MVI-E, 2 0       MOV-D,A 1 0      SUB-B 1 0        RET 1 0          PUSH-PSW 1 0
RAR 1 0          MOV-E,B 1 0      SUB-C 1 0        JZ 3 2           ORI 2 0
JRNZ 2 1         MOV-E,C 1 0      SUB-D 1 0        CB 2 0           RST-6 1 0
LXI-H, 3 0       MOV-E,D 1 0      SUB-E 1 0        CZ 3 2           RM 1 0
SHLD 3 0         MOV-E,E 1 0      SUB-H 1 0        CALL 3 2         SPHL 1 0
INX-H 1 0        MOV-E,H 1 0      SUB-L 1 0        ACI 2 0          JM 3 2
INR-H 1 0        MOV-E,L 1 0      SUB-M 1 0        RST-1 1 0        EI 1 0
DCR-H 1 0        MOV-E,M 1 0      SUB-A 1 0        RNC 1 0          CM 3 2
MVI-H, 2 0       MOV-E,A 1 0      SBB-B 1 0        POP-D 1 0        FD 2 0
DAA 1 0          MOV-H,B 1 0      SBB-C 1 0        JNC 3 2          CPI 2 0
JRZ 2 1          MOV-H,C 1 0      SBB-D 1 0        OUT 2 0          RST-7 1 0
DAD-H 1 0        MOV-H,D 1 0      SBB-E 1 0        CNC 3 2          END 0 0
LHLD 3 0         MOV-H,E 1 0      SBB-H 1 0        PUSH-D 1 0
DCX-H 1 0        MOV-H,H 1 0      SBB-L 1 0        SUI 2 0
INR-L 1 0        MOV-H,L 1 0      SBB-M 1 0
DCR-L 1 0        MOV-H,M 1 0      SBB-A 1 0
MVI-L, 2 0       MOV-H,A 1 0      ANA-B 1 0
CMA 1 0          MOV-L,B 1 0      ANA-C 1 0
JRNC 2 1         MOV-L,C 1 0      ANA-D 1 0
LXI-SP, 3 0      MOV-L,D 1 0      ANA-E 1 0
STA 3 0          MOV-L,E 1 0      ANA-H 1 0
INX-SP 1 0       MOV-L,H 1 0      ANA-L 1 0
INR-M 1 0        MOV-L,L 1 0      ANA-M 1 0
DCR-M 1 0        MOV-L,M 1 0      ANA-A 1 0
MVI-M, 2 0       MOV-L,A 1 0      XRA-B 1 0
STC 1 0          MOV-M,B 1 0      XRA-C 1 0
JRC 2 1          MOV-M,C 1 0      XRA-D 1 0
```

Abb. 2.5.1-2 8080 Datensatz für den Disassembler

erhöht, und anschließend wird aus der relativen Adresse eine absolute Adresse berechnet. Dabei sind zwei Fälle zu unterscheiden. Falls B kleiner ist als 8ØH (im BASIC durch & gekennzeichnet, daß es sich um eine sedezimale Konstante handelt), so kann die Adresse direkt aus der Addition des neuen Standes des Programmzählers und der Konstante B berechnet werden. Andernfalls, wenn B größer oder gleich 8ØH ist, wird (B-256) zu dem Programmzähler hinzugezählt. Die nächste Zeile dient dem Fehlerfall, wenn nämlich das Ende der File erreicht ist, und B den Wert 999 hatte.

In 660 wird mit Hilfe des Unterprogramms 11000 die gewonnene Adresse in die Symboltabelle eingetragen. Auf ähnliche Weise erfolgt der Eintrag für die anderen zwei Fälle. Der Anwender kann die Adreßarten noch erweitern, falls er z.B. einen anderen Prozessor implementieren möchte, der keine von den hier verwendeten Adressierarten besitzt.

In Zeile 1000 beginnt PASS2 des Disassemblers, der der Erzeugung eines Listings dient. Zunächst wird in Zeile 1030 der alte Programmzähler zurückgespeichert, und der Symboltabellenpointer S2 wird auf 0 gesetzt. Falls die Quelle eine Diskfile war, ist es dann erforderlich, diesen zunächst zu schließen und dann wieder zu öffnen, um den Datenpointer wieder auf den Anfang zu setzen. Das gleiche geschieht mit dem Speicherzeiger A.

Die Symboltabelle muß nun aufsteigend sortiert werden. Dazu dienen die Zeilen 1100 bis 1140, die einen „Blubble Sort" darstellen.

Der nun folgende Teil ist im Prinzip genau so aufgebaut, wie der Pass 1. Es erfolgt aber kein Symboleintrag, sondern zu Beginn jeder Zeile wird festgestellt, ob eine Markengenerierung erforderlich ist. Dies geschieht mit dem Unterprogramm 13000. In M$ wird der mnemotechnische Code eines Befehls abgespeichert, der über das Array M$ ( ) verfügbar ist. Die Adresse wird in AD$ gespeichert. In B2$ und B3$ wird gegebenenfalls eine zusätzliche Daten- oder Adreßkonstante abgelegt. Auf 2010 erfolgt die Ausgabe der Zeile mit dem Aufruf für das Unterprogramm 12000. Dabei werden zwei unterschiedliche Listings erzeugt. Auf 12020 erfolgt die Ausgabe auf die Konsole bzw. Drucker mit einer „Printusing"-Anweisung. Dabei wird auch der Objektcode mit ausgegeben. Auf 12050 erfolgt die Ausgabe auf den Disk, die mit einer Print-Anweisung erfolgt. Die Zeile wird dabei so ausgegeben, daß sie von TDL Assembler direkt verstanden wird.

Abb. 2.5.1-3 zeigt das Beispiel eines disassemblierten Programms. Dabei wird am Anfang noch ausgegeben, wo Adressen erkannt werden. Ferner wird die sortierte Symboltabelle ausgegeben. *Abb. 2.5.1-4* zeigt die Diskfile, wie sie vom Disassembler abgelegt wird.

2.6 Testhilfen für Software

Wenn angenommen wird, daß ein Programm sofort nach dem ersten Start läuft, so ist das meist ein Irrtum. Um Fehler in einem Programm zu finden, gibt es verschiedene Hilfen. Es handelt sich dabei entweder um eine Hardware-Schaltung, die es ermöglicht, durch Programme in Einzelschritten durchzukommen oder um Testprogramme, die andere Programme testen können. Im folgenden sollen solche Testprogramme behandelt werden.

Eine einfache Art von Testbefehlen ist in den meisten Monitorprogrammen enthalten. Mit diesen Befehlen ist es z.B. möglich, Breakpoints (Haltepunkte) zu setzen. Wird dann das zu testende Programm gestartet, so wird wieder der Monitor aufgerufen,

```
PASS 1
ABSADR          C0EE
RELADR:         C018
ABSADR          C10B
ABSADR          C229
RELADR:         C01C
ABSADR          C03F
RELADR:         C03A
ABSADR          C08C
ABSADR          0080
RELADR:         C024
ABSADR          C0CB
RELADR:         C06A
PASS 2
OK SYMBOLTABELLE SORTIERT
0000            0080            C018            C01C            C024
C03A            C03F            C06A            C08C            C0CB
C0EE            C10B            C229
 C000   21 7C 00                    LXI H,      0007CH
 C003   F9                         SPHL
 C004   EB                         XCHG
 C005   F3                         DI
 C006   CD EE C0                   CALL        MC0EE
 C009   97                         SUB A
 C00A   D3 03                      OUT         00003H
 C00C   DB 34                      IN          00034H
 C00E   E6 40                      ANI         00040H
 C010   28 06                      JRZ         MC018
 C012   C3 0B C1                   JMP         MC10B
 C015   CD 29 C2                   CALL        MC229
 C018   3E D0           MC018:     MVI A,      000D0H
 C01A   D3 30                      OUT         00030H
 C01C   DB 30           MC01C:     IN          00030H
 C01E   1F                         RAR
 C01F   38 FB                      JRC         MC01C
 C021   F3                         DI
 C022   3E 10                      MVI A,      00010H
 C024   21 80 00        MC024:     LXI H,      00080H
 C027   F9                         SPHL
 C028   F5                         PUSH PSW
 C029   44                         MOV B,H
 C02A   CD 3F C0                   CALL        MC03F
 C02D   20 0B                      JRNZ        MC03A
 C02F   F1                         POP PSW
 C030   F5                         PUSH PSW
 C031   44                         MOV B,H
 C032   1E 01                      MVI E,      00001H
 C034   CD 8C C0                   CALL        MC08C
 C037   CA 80 00                   JZ          M0080
 C03A   F1              MC03A:     POP PSW
 C03B   EE 10                      XRI         00010H
 C03D   18 E5                      JMPR        MC024
 C03F   CD CB C0        MC03F:     CALL        MC0CB
 C042   D3 34                      OUT         00034H
 C044   16 98                      MVI D,      00098H
 C046   E6 10                      ANI         00010H
 C048   3E 7F                      MVI A,      0007FH
 C04A   D3 04                      OUT         00004H
 C04C   3E 0F                      MVI A,      0000FH
 C04E   28 1A                      JRZ         MC06A
```

Abb. 2.5.1-3 Beispiel eines disassemblierten Programms

```
A>TYPE RDO.$$1

        LXI H,  0007CH
        SPHL
        XCHG
        DI
        CALL    MC0EE
        SUB A
        OUT     00003H
        IN      00034H
        ANI     00040H
        JRZ     MC018
        JMP     MC10B
        CALL    MC229
MC018:  MVI A,  000D0H
        OUT     00030H
MC01C:  IN      00030H
        RAR
        JRC     MC01C
        DI
        MVI A,  00010H
MC024:  LXI H,  00080H
        SPHL
        PUSH PSW
        MOV B,H
        CALL    MC03F
        JRNZ    MC03A
        POP PSW
        PUSH PSW
        MOV B,H
        MVI E,  00001H
        CALL    MC08C
        JZ      M0080
MC03A:  POP PSW
        XRI     00010H
        JMPR    MC024
MC03F:  CALL    MC0CB
        OUT     00034H
        MVI D,  00098H
        ANI     00010H
        MVI A,  0007FH
        OUT     00004H
        MVI A,  0000FH
        JRZ     MC06A
.END
```

Abb. 2.5.1-4 Diskfile wie sie vom Disassembler abgelegt wird

falls ein solcher Breakpoint erreicht wurde. Dann ist es möglich, Register und Speicherzellen zu betrachten und gegebenenfalls das Programm bis zu einem neuen Breakpoint weiterlaufen zu lassen. Sehr wesentlich ist die höchste Anzahl von Breakpoints, die gleichzeitig gesetzt werden kann. Zum vernünftigen Arbeiten sollen mindestens zwei Breakpoints möglich sein. Etwas komfortabler wird das Ganze wenn Einzelschritte ausgeführt werden können. Dabei wird meistens wie bei der Technik mit Breakpoints verfahren. Das heißt, es wird ein Befehl RST 7 an die Stelle geschrieben, an der der Breakpoint auftreten soll. Im Falle des Betriebs mit Einzelschritten wird das Setzen des Breakpoints vom Programm automatisch vorgenommen. Nach jedem Schritt werden dann die Inhalte der Register des Prozessors sichtbar. Der nächste Befehl wird jeweils mit Tastendruck ausgeführt. Nachteilig an diesem Verfahren ist, daß es nicht in einen ROM angewendet werden kann, da der Befehl RST 7 in den Speicher geschrieben werden muß. Ferner können bei manchen Programmen auch Schwierigkeiten entstehen, daß es Manipulationen mit Stackpointern vornimmt. Z.B. wird dabei ein im Stack weiter zurückliegender Wert erneut verwendet. Wird nämlich ein Befehl RST 7 ausgeführt, so wird auf dem Stack die Rückkehradresse abgelegt. Dabei wird der an dieser Stelle im Stack liegende ältere Wert zerstört. Will ein Anwender aber auch ältere Werte des Stack verwenden, dann ist dieser Wert nicht mehr vorhanden und es kommt zu Fehlern. Das gleiche würde auch geschehen, wenn bei einem solchen Programm mit INTERRUPT gearbeitet werden würde. Beim Programmieren sollte also diese Technik vermieden werden, da die erzeugten Programme nicht fähig für INTERRUPT sind. Manchmal läßt es sich aber nicht vermeiden, z.B. in zeitkritischen Teilen, in denen bekanntlich mit allen „Tricks" gearbeitet werden darf.

Im folgenden soll eine Testhilfe für den 8080 und Z80 besprochen werden, die allerdings nur auf dem Z80 lauffähig ist, die Programme in ROMs und auch nicht interruptfähige Programme aufzeichnen (tracen) kann.

2.6.1 Softwaretracer

Der Softwaretracer hat einen Umfang von etwa 3 KByte. Er ist in *Abb. 2.6.1-1* im Relocating Format abgelegt. Dieser Tracer ist nur im RAM lauffähig.

2.6.1.1 Befehle des Tracers

Nach dem Starten des Tracers auf seiner ersten Adresse meldet er sich mit einem Registerausdruck. Dann wird ein Befehl erwartet. Nachstehend sind die Befehle zusammengestellt.

Padrcr
Setzen des Programmzählers. Der Programmzähler wird auf den Wert adr gesetzt. Der Abschluß der Befehlseingabe erfolgt hier mit cr.

Beispiel: P2ØØØcr setzt den Programmzähler auf die sedezimale Adresse 2ØØØH.

, (Komma)
Bei der Eingabe von , (Komma) wird der Befehl ausgeführt, auf den der Programmzähler zeigt. Dabei ist der Befehl selbst auch auf dem Bildschirm sichtbar. Nach der Ausführung des Befehls sind alle Register auf dem Bildschirm mit den aktuellen Werten belegt.

```
',MAIN.7F
\03.PROG.010DBB.DATA.020000.BLNK.030000A2
;1800000140C35A07C309F0C30300F0C312F0C30FF07900FE0AC8F5DB71
;1800150100FFE60128FAF12FF60080D3FFE67FD3FFF60080D3FF2FC9EB
;18002A010079FE0AC8F53EC0D300FE3E17D3FD3E0ACD807F00F1D3FCB7
;18003F0102F5DBFDCB47CA400001F1C501FC0004C24C10000CC24C00CD
;1800540100C1FE0DC0F53E50CD817F00F1C5D5E521910005E2B50EA8
;1800690111000CC26A001DC26A100015C26A00360123003601E1D1C197
;18007E0110C9E521910086770280BE002B7EC601772300E1C9010179ED
;1800930100C321E079FE0AC8FE000DC221E03EA0C32100E079F5F5CDA7
;1800AB010221E0F1FE0DC2B500003E0DCD21E0F1C97900F5C39BE979C8
;0500BD0100F5C397E905
;1800E10120F53A9D0AB72004F144C30300F1C30C007900C30CE0CD097C
;1800F60100E0B728FAC9C315E0000600E5D17EE6DFFE20DDCA74017E00
;18010B0110FECBCA6E01FEEDCA8262017EFEC3CAAB0110FECDCAAB0129
;1801200104E6EFFE22CAAB01FE202ACAAB01E6CFFE0140CAAB01E6C782
;1801350110FEC2CAAB01FEC4CA80AB017EE6F7FE10CA80AC01FED3CAB8
;18014A0181AC01E6E7FE20CAAC0201E6C7FE06CAAC0110FEC6CAAC0197
;18015F0140C3AD01237EE6C7FE2443CAAA01C3AC01C390AC01C3AA01D0
;1801740104237EFECBCA7101FE2021CAAA01E6FEFE3440CAAB01E6F86A
;1801890110FE70CAAB017EE6CF10FE06CAAA01E6C7FE2002CAAA017EED
;19019E0101D640E687FE06CAAB2001C3AC010404040408EBC9ED43A20A11
;1801B40124ED53A40A22A60AF510C1ED43A00A08D9ED4043AA0AED5369
;1801C90190AC0A22AE0AF5C1ED4243A80AED5732B60A10ED5F32B70A9E
;1801DE0122DD22B00AFD22B20A422A9E0AC9FD2AB20A20DD2AB00A3A77
;1801F30184B60AED473AB70AED124FED4BA80A229E0A10C5F12AAE0AD6
;1802080122ED5BAC0AED4BAA0A08D908ED4BA00AC5F1402AA60AED5BEE
;18021D0180A40AED4BA20AC9AF1006041147021218101FCD5CDFE00D9
;1802320100D17E13122310FACD88EA01ED73B80AED7B80B40A0000000A
;1802470111100ED73B40AED7BB8200ACDB001C97E23ED4073B80AED7B73
;18025C0184B40AE5ED73B40AED407BB80A2600E6386F00C9232323ED08
;1802710144738B0AED7BB40AE522ED73B40AED7BB80A002B7E2B6E67DD
;1802860122C92AA60AC92AB00A21C92AB20AC9ED73B8100AED7BB40AFC
;18029B0111E1ED73B40AED7BB8000AC9237E23666FC900237E234FB71B
;1802B00140FAB702060009C90604FF09C9ED4BA20A0520ED43A20AC2E8
;1802C50180AB0223232C97EE638B11F6C232D802ED4BA0080AC5F1006D0263
;1802DB0102232323C97E32E90221ED4BA00AC5F100A500022323323C9A9
;1802F001027EE638F6C232FE0221ED4BA00AC5F100930002223C97EE6CF
;1803050108183F6C2321103ED4B84A00AC5F100AB02230023C97EFEC9A4
;18031A0142CA9302FECDCA6D0210FEC3CAA502FE18CAB0AB02FE10CAFE
;18032F0184B02FEE9CA8702E608E7FE20CA0303E6C710FEC2CADF024F
;1803440110FEC4CACA02FEC7CA845402FEC0CAF0027E10FEEDC26903AE
;1803590102237E2BFE4DCA930212FE45CA9302C32402007EE6DFFEDD58
;18036E0140C22402237E2BFEE941C224027EFEDDCA8B2002C38F027ED0
;1803830104B7C84F23CDE1001811F67CCDB9037DC3B90003F51F1F1F4C
;1803980120 1FCD9D03F1E60FC60830FE3ADAAB03C60740C3AB034FC36C
;1803AD0184E1000E0DCDE1000E240AC3E100CD94030E2020C3E100C50E
;1803C2012178CD9403C179C3B90203E5210000CD0600204FCDE100CDA7
;1803D70190E403DAF403292929002956F18EBD630D800FE173FD8FEF2
;1803EC01000A3FD0D607FE0AC90279E5C1E1C9CD02042047CD02044F0A
;1904010110C9C5CD110487878710874 7CD110480C1C948CD0600CDAB036C
;19041 70100FE3038F6FE3A380200D607E60FC9E50E0C48CDE10021EF0558
;18042D0142CD8203C1C5CDC10348211B06CD8203E1E540CDFE007E23BD
;1904420120C5CDB903C110F721922806CD8203CDAF0348CDAF03213A0690
;1804580148CD82032AA00A7DCD89B2057CCDE9032AA2200ACD8C032A12
;18046D0192A40ACD8C032AA60A49CD8C03217F06CD8220032AA80A7DEA
;1904820144CDB2057CCDB9032A92AA0ACD8C032AAC0A48CD8C032AAE0A61
;1804980149CD8C03CDAF03CDAF240321B306CD8203ED404B800AC5CDC4
;1804AD0184C103C10ACDB903ED444BB20AC5CDC103C1200ACDB9032A6E
;1804C20189A60A7ECDB9033AB6240ACDB9033AB70ACD80B9030E20CD3B
;1804D70188E100ED4B840AC5CD80C103D11B1B0603C5001A4F131A4725
;1804EC012013CDC103C110F3CD92AF03CDAF033A9C0A20B7CA6B053EB0
;180501012401329D0A213807CD9182032AA00A7DCDB910037CCDB903B1
;18051601 24ED4BA20ACDC103ED484BA40ACDC103ED4B90A60ACDC1036B
```


Abb. 2.6.1-1 Softwaretracer Object Code

```
;18052B0148214D07CD8203ED4B91B00ACDC103ED4BB2200ACDC103ED02
;19054001494BB40ACDC10321552107CD8203C1C5CDC108030E20CDE100D3
;1805560110E1E5CDFE007E23C541CDB903C110F7CDAF1003AF329D0ADC
;19056B01443ABA0AB7C27A052190EB06CD8203E1C9F54821B506CD820333
;1805810120F121BB0A5E23562308EBF5D5CD9E05D1F110EB3DC28505FD
;180596014821EB06CD8203E1C910E5C1CDC1030610C5207ECDB903238A
;1805AB0108C110F7CDAF03C9CB217FCDE305CB77CDE32005CDDB05CB70
;1905C0012467CDE305CDDB05CB2157CDE305CB4FCDE30805CB47CDE3059E
;1805D601100E20C3E100F50E2040CDE100F1C9F50E2A0020020E20CD15
;1805EB0180E100F1C952444B20005A383020534F46540057415245543A
;18060001005241434552205631002E31203739303232003620 0D0A0DD0
;18061501000A50433A20002020002020202020424546003A200020208E
;18062A01002020202020202020002020202020202000002053 5A78484A
;18063F010078504E43204120200042202043202044200020452020485 2
;180654010020204C202020202000535A784B78504E43002041272042B1
;18066901002720432720442720004527204827204C27000D0A0D0A2040
;18067E01000020202000204958002020202020204959002020202020 40
;180693010028484C29204920200052202053502020200 02D322020206C
;1806A801003020202020 2B320D000A0D0A20004D454D002E2030202041
;1806BD01003120203220203320002034202035202036002020372020F8
;1806D201003820203920204120002042202043202044002020452020 8F
;1806E70100460D0A002020202000202020202020202000202020202 07D
;1806FC01002020202020202020002020202020202020002020202020 45
;18071101002020202020202020002020202020200D42004546454 84C5C
;18072601003E20002020202020002020202020202020002000464142D3
;18073B0100434445484C20002700464142434445484C002000495849CB
;18075001005953502000424546102000 31BA0D21010140229000AF3289
;18076501929C0A329D0A328F0A120E0CCDE10021290740CD8203AF3201
;18077A0182BA0A210000CD240448CD0600CDAB03FE4440CAEE07FE2C09
;18078F0142CA4808FE2ECA240810FE3BCA3608FE50CA804E08FE4ECA7E
;1907A401845608FE55CA7308FE214DCA8A08FE47CABF0808FE53CA660991
;1807BA0110FE54CA760AFE42CA92C707C37F07CD060040CDAB03FE59E2
;1807CF0149C27F07CD0600CDAB0903FE45C27F07CD062000CDAB03FE32
;1807E401200DC27F0731E01FC31203E0CD0600CDAB0310FE30DA7F07B3
;1807F90110FE38D27F07E607E540211408074F060009005E23666B2221
;19080E018A0D00E1C37F070F00AA2A0092009600A400A8B700BC00C10084
;18082401123E01329C0ACD170344CD2404AF329C0AC3807F073E0132B1
;18083901929D0ACD1703CD240424AF329D0AC37F07CD901703C37F07DC
;18084E0142CDCB03C5E1C37F0744CDCB03C5CD1703CD902404CD0900DF
;1808630140C26F08C10B78B12022EDC37F07C1C37F0740CDCB03C5CD1F
;19087801921703CD0900C26F0804C10B78B1C27608C3887F07E5CD0600E9
;18088E0142CDAB03FE43C2A30821AF32BA0A0E0CCDE11000E1C37F071E
;1808A301443ABA0A3C32BA0A07085F160021B90A19CD80CB0379E6F0DD
;1808B80104772370E1C37F07AF44328F0AE5CDCB0360206922F208FEAE
;1908CD01002C2824FE0D2807E1482A9E0AC37F073EC3083238002134094A
;1908E30108223900CDEA01ED7388B80AED7BB40AC3000800AF5721900A84
;1808F90140CDCB035F7123702304147AFE05328F0A2800CA7BFE0D208D
;18090E0122EA21900ADD21980A004E2346230ADD770000DD233EFF02F2
;19092301001520F03EC33238004121340922390 0C3DB0808E32B229E0AAA
;1809390104E33333ED73B40AED497BB80ACDB0013A8F100AB7CAD40809
;18094E01225721900ADD21980A004E234623DD7E00DD0023021520F35D
;1809630144C3D408E5CD0600CD84AB03FE5221A00A3E2020CA180ACD8F
;1809780181CB036069FE0DCAA610097ECD94030E2DCD90E100CDCB03C4
;18098D0100FE2C28197123FE2E122803C37C09CDAF0310E5C1CDC103DB
;1809A201023E0118D6E1C37F072223C37C092049B60A10012052B70AE9
;1809B70110012041A00A02204284A20A022043A20A01202044A40A0231
;1809CC01212045A40A012048A6080A02204CA60A01204046A00A0158F5
;1809E1014249B00A005949B20A10005053B40A00274180A80A022742E4
;1909F60184AA0A022743AA0A01212744AC0A022745AC080A012748AE0AF8
;190A0C011002274CAE0A01274684A80A0100CD0600CD88AB0357CD0600EE
;180A220141CDAB035F0E20CDE1210001AE090AB7CAA600096F030A67CE
;180A370110ED52CA420A030303000318EC030A6F030A0067030AB7205D
;180A4C0104144E2346CDC1030E242DCDE100CDCB0370102B71C3A609FB
;180A6101013D4F0600097ECD9409030E2DCDE100CDCB108371C3A6097E
;190A760148CD0600CDAB03FE4E0420083E01329C0AC3887F07AF329C0AE9
;040A8C0140C37F07DC
;00075A019E
```

zu Abb. 2.6.1-1

Nexprcr
Es werden mehrere Befehle hintereinander ausgeführt. Dabei ergibt der Teil expr die Anzahl der auszuführenden Befehle an. Mit 1FH werden 1FH-Schritte ausgeführt, das heißt dezimal 31 Befehle. Die Registerinhalte erscheinen nach jedem Befehl.

Uexprcr
Wie bei Nexprcr, nur daß die Registerinhalte nicht nach Ausführung von jedem Befehl erscheinen. Diese Art von Befehl wird angewendet, wenn schnell über ein Programmgebiet hinweg-„getraced" werden soll. Beide Befehle können mit CTRL C unterbrochen werden. Dann werden die aktuellen Registerinhalte angezeigt.

MSadrcr
Zusätzlich zu den angezeigten Registern wird ein Speicherbereich von 16 Byte mit angezeigt, der bei der durch adr bestimmten Adresse anfängt. Dabei wird die Zahl auf einen glatten Wert gerundet. MS1234cr zeigt bei jedem Registerausdruck den Bereich 123ØH bis 123FH. Es können mehrere Speicherbereiche definiert werden und zwar in beliebiger Reihenfolge. Diese Bereiche werden dann immer mit ausgegeben. Die höchste Anzahl ist aber 256!

MC
Löschen der Simultanausgabe von Speicherbereichen.

Gadrcr
Starten eines Programms in Echtzeit. Falls in dem Programm in der gleichen Aufrufstufe ein RST 7 vorkommt, so wird wieder in den Tracer zurückgesprungen (die Wirkungsweise ist wie bei dem Aufruf eines Unterprogramms).

Gadr, adr1, adr2, adr3, adr4cr
Ein Programm wird bei der Adresse adr gestartet. Der Tracer meldet sich wieder, wenn entweder eine der vier Zusatzadressen (adr1 . . . adr4) oder ein RET-Befehl erreicht wird. Dabei wird das Programm in Echtzeit gestartet. Ein Breakpoint darf nur auf den Anfang eines Befehls gesetzt werden.

Beispiel: G1ØØØ,1ØØ3cr
Speicher 1ØØØ

0	1	2	3	4	5	6	7	8
23	ØE	Ø1	C2	ØØ	Ø1	CA	ØØ	Ø2

Bedeutung:

1ØØØ:	23	INX H
	ØE Ø1	MVI C,1
	C2 ØØ Ø1	JNZ 1ØØH
	CA ØØ Ø2	JZ 2ØØH

Das Programm wird bei 1ØØØH gestartet. Auf Adresse 1ØØ3H wurde ein Breakpoint gesetzt, so daß das Programm beim Erreichen dieser Stelle wieder in den Tracer zurückkehrt.

SMadrb
Mit diesem Befehl ist es möglich, Speicherinhalte zu modifizieren, um z.B. Programme in sedezimaler Schreibweise einzugeben. Bei der Eingabe des Befehls wird die auf adr befindliche Zeile dargestellt. Durch Eingabe eines neueren Wertes mit nachfolgendem Blank (Leerzeichen) wird der neue Wert in den Speicher übernommen. b bedeutet Eingabe eines Blanks, so daß nach Eingabe der Adresse zur Anzeige des ersten Wertes ein Zeichen Blank ebenfalls eingegeben werden muß. Wird kein Wert angegeben, so kann mit Komma zur nächsten Zelle weitergegangen werden. Wird an Stelle des Blank ein Punkt eingegeben, so wird in die nächste Zeile geschaltet und die neue aktuelle Adresse wieder ausgegeben.
Beispiel: SM1ØØØ 23-2Bcr

Die Zelle 1ØØØH enthält am Anfang den Wert 23H und wird auf 2BH verändert.

SRreg
Hiermit ist es möglich, Registerinhalte zu verändern. Mit reg sind folgende Zeichenfolgen gemeint:

Ab	Bb	Cb	Db	Eb	Hb	Lb	Fb
A'	B'	C'	D'	E'	H'	L'	F'
IX	IY	SP	Ib	Rb			

Dabei steht b immer für Blank (Leerzeichen).
Beispiel: SRA ØØ-22cr
Der Inhalt des Akkumulators wird auf 22H gesetzt.

TN
Ein angeschlossener Drucker wird eingeschaltet, um einen zusätzlichen Ausdruck des Trace zu ermöglichen. Dabei kann mit einem Drucker gearbeitet werden, der 80 Stellen aufweist, und der für jeden Schritt des Trace eine Kompaktzeile des Registerinhalts ausgibt.

TO
Der Trace des Druckers wird wieder ausgeschaltet.

. (Punkt)
Einzelschrift mit Druckausgabe. Ein Schritt wird ausgeführt und zusätzlich eine Zeile Kompaktausdruck auf dem Drucker ausgegeben.

; (Semikolon)
Hier wird auch ein Schritt ausgeführt, doch wird der Bildschirminhalt auf dem Drucker ausgegeben.
Abb. 2.6.1-2 zeigt einige Beispiele eines Vorgangs „Trace".

2.6.1.2 Anpassung des Tracers an vorhandene Systeme

Abb. 2.6.1.2-1 zeigt den Ausdruck von einem Assembler für den Anfang der Trace. Um den Tracer an ein vorhandenes System anzupassen, genügt es vier Sprungbefehle zu ersetzen. Der erste Sprungbefehl führt auf ein Programmteil mit dem Namen COS. Dies ist die Ausgaberoutine, die die Aufgabe hat, ein im Register C befindliches Zeichen

auf der Konsole auszugeben. CI hat die Aufgabe, in das Register A zu lesen. CSTS prüft den Status der Konsole, indem es den Wert ØFFH in das Register A legt, falls ein Zeichen an der Konsole eingegeben wurde. Wenn das nicht der Fall ist, wird das Zeichen Ø eingelesen. LO schließlich dient zur Druckerausgabe, das Zeichen C wird an den Drucker ausgegeben. Das Zeichen ØCH wird in die Konsole ausgegeben, um den Bildschirm zu löschen.

```
RDK Z80 SOFTWARETRACER V1.1 790226

PC: BE61          BEF: 31 00 01

 SZXHXPNC A  B  C  D  E  H  L      SZXHXPNC A' B' C' D' E' H' L'

  * *  *  4B 41 20 52 44 FF FF      * *  ** 45 0A 0D 50 4F 3A 52

 IX       IY      (HL) I  R  SP   -2   0    +2

 4420 5A 2053 08 07 3B 35  0928 88E9 4120 F6C5


BEFEHL> FF01E
RDK Z80 SOFTWARETRACER V1.1 790226

PC: F01E          BEF: C3 BE F6

 SZXHXPNC A  B  C  D  E  H  L      SZXHXPNC A' B' C' D' E' H' L'

  * *  *  4B 41 20 52 44 FF FF      * *  ** 45 0A 0D 50 4F 3A 52

 IX       IY      (HL) I  R  SP   -2   0    +2

 4420 5A 2053 08 07 3B 35  0928 88E9 4120 F6C5
```

Abb. 2.6.1-2 Beispiel eines Vorgangs „Trace"

```
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 0928 BEF F01E  C3 BE F6
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 0928 BEF F6BE  F5
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 0926 BEF F6BF  D5
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 0924 BEF F6C0  C5
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 0922 BEF F6C1  F5
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 0920 BEF F6C2  CD B9 F5
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 091E BEF F5B9  C5
FABCDEHL 52 4B 4120 5244 FFFF IXIYSP 4420 2053 091C BEF F5BA  21 FF FF
```

zu Abb. 2.6.1-2

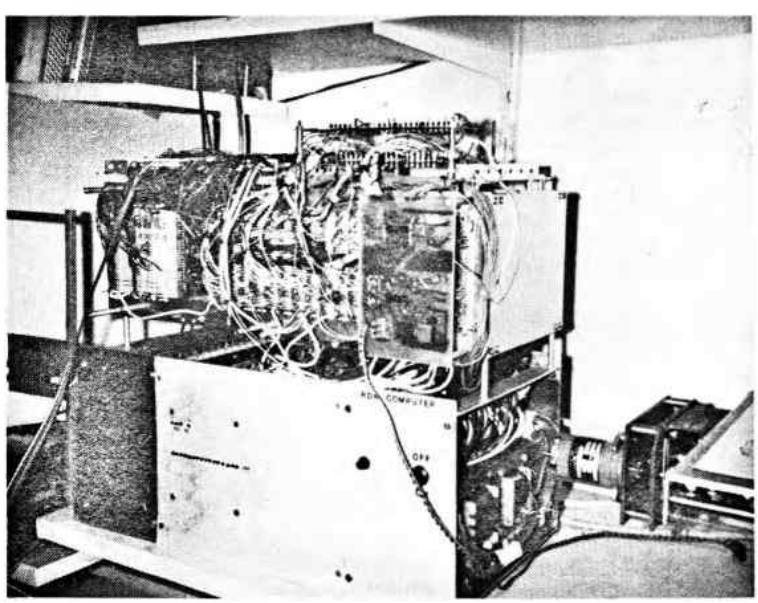
```
0000'   C3 075A'      START:JMP MAIN
0003'   C3 F009       COS:JMP 0F009H
0006'   C3 F003       CI: JMP 0F003H
0009'   C3 F012       CSTS:JMP 0F012H
000C'   C3 F00F       LO: JMP 0F00FH
```

zu Abb. 2.6.1-2

2.6.1.3 Funktionsweise des Tracers

Der Tracer selbst belegt etwa 3 KByte Speicherplatz. Die meisten Variablen sowie der Stack sind hinter dem Tracer angeordnet. Insgesamt wird also ein Speicherplatz von 4 KByte vom Tracer benötigt. Dabei wird außerhalb dieses Bereiches kein Speicherplatz beansprucht. Daher steht der restliche Platz voll für das Programm des Benutzers zur Verfügung. Der Tracer arbeitet wegen einer möglichst hohen Geschwindigkeit für die Aufzeichnung mit einer selbst modifizierenden Technik. Er kann daher nicht in ROMs laufen. Der Tracer kann jederzeit wieder vom Anfang an gestartet werden. Dabei besteht nicht die Gefahr, daß er wegen der Modifizierung nicht wieder anläuft, da er die Bereiche immer wieder initialisiert.

Bei dem Tracer handelt es sich um eine Art EMULATOR. Dabei werden zur Ausführung eines Befehls des Benutzerprogramms keine Breakpoints gesetzt. Der Tracer wirkt vielmehr wie eine eigene CPU, die einen Befehl vom Benutzerprogramm holt und dann ausführt. Dafür müßte jeder Befehl analysiert werden, und mit den Speicherzellen, die als Register dienen, ausgeführt werden. Da aber die Maschine, die emuliert werden soll, auf ein- und derselben Maschine läuft, die schon in der Lage ist, diese Befehle auszuführen, ist es nicht nötig, jeden einzelnen Befehl genau auszuführen. Es genügt, wenn die Zugehörigkeit zu einer Befehlsgruppe festgestellt wird. Dann können die meisten Befehle dadurch ausgeführt werden, daß sie in einen Speicherbereich geschrieben werden. Der Tracer springt dann einfach dorthin, um den Befehl auszuführen. Dabei müssen vor Ausführung des Befehls zunächst alle Register geladen und dann wieder zurückgespeichert werden. Dies geht aber nicht mit allen Befehlen. Betroffen sind alle Befehle, die den Programmzähler beeinflussen. Bei Ausführung dieses Befehls würde dann der Prozessor einfach direkt in das Benutzerprogramm springen. Der Tracer hätte keine Kontrolle mehr über den Ablauf des Benutzerprogramms. Die Befehle werden in dem Tracer aussortiert und direkt durch Setzen eines Pseudoprogramm-Zählers ausgeführt. Bei den Befehlen CALL und RET wird zusätzlich die Verwaltung des Stacks vorgenommen.



3 Software, verschiedene Programme

3.1 Lader für das TDL Relocating Format

An dieser Stelle soll ein Programm besprochen werden, das die Aufgabe hat, das schon benutzte Relocating Format in den Programmspeicher zu laden. Dabei arbeitet das Programm mit dem CP/M-Betriebssystem und Diskette. Es kann aber leicht an ein System angepaßt werden, das nur mit Konsole arbeitet.

3.1.1 CP/M-Betriebssystem

Zunächst soll das CP/M-Betriebssystem näher besprochen werden. Es handelt sich hier um ein Betriebsprogramm, das die Aufgabe hat, das Arbeiten mit Disketten zu ermöglichen. Das System CP/M ist von DIGITAL RESEARCH sowohl für große als auch für kleine Disketten lieferbar und benötigt einen minimalen Arbeitsspeicherraum von 16 KByte einschließlich des Platzes für das Anwenderprogramm. Dabei können auf einer großen Floppy etwa 250 KByte und auf einer MINI-Floppy etwa 90 KByte untergebracht werden, normale Aufzeichnungsdichte vorausgesetzt. Bei doppelter Dichte sind es 180 KByte und bei weiterer Verdopplung durch die doppelte Spurdichte sogar 360 KByte auf der MINI-Floppy. Eine Floppy ist in Sektoren und Spuren unterteilt. Bei der großen Floppy sind es 76 Spuren und 26 Sektoren. Abb. 3.1.1-1 zeigt den Aufbau einer Floppy. Ein Sektor enthält bei einfacher Dichte 128 Byte. Die Position eines Sektors wird mit Hilfe eines Indexloches bestimmt. Dies kennzeichnet den Sektor 0. Wird bei Disketten der Sektor durch die Software definiert, dann werden die anderen Sektoren durch einen Kennzeichnungsblock bestimmt, der auf der Diskette steht. Bei den Disketten mit Orientierung durch Hardware wird die Position durch weitere Löcher auf der Diskette, den sogenannten SECTOR HOLES bestimmt. Die durch die Löcher bewirkten Impulse werden abgezählt, wobei das Rücksetzen durch den Index-Impuls ausgelöst wird.

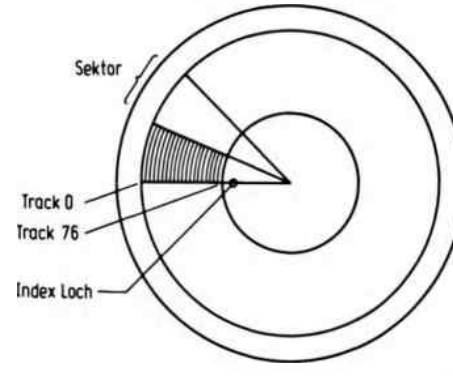


Abb. 3.1.1-1 Anordnung von Sektoren auf einer Floppy

Daten und Programme, die auf der Platte stehen, müssen irgendwelche Positionen haben, die dem Benutzer bekannt sein müssen. Um nicht jedem Programm von Hand mit Bleistift und Papier die entsprechenden Sektoren und Spuren (Tracks) zuweisen zu müssen, gibt es Disketten-Betriebssysteme. Diese arbeiten sehr unterschiedlich.

FDOS ist z.B. ein solches Betriebssystem. Einem Programm kann ein beliebiger Name zugewiesen werden, unter dem es später erreichbar ist, ebenso wie die Daten. Dazu wird ein sogenanntes „Directory" angelegt.

Das „Directory" ist ein reservierter Platz von mehreren Sektoren und Tracks auf der Diskette. Dort steht der Name des Programms im 7-Bit-Code (ASCII) mit 5 Buchstaben und dazu die Anfangsadresse des Programms auf der Diskette. Dazu wird der erste Sektor und Track, bei dem das Programm beginnt, abgelegt. Ferner wird die genaue Anzahl der Sektoren dieser „File" (Datei) festgehalten. Einer Datei können dennoch Attribute zugewiesen werden. Solche Attribute sind z.B. Schreibschutz oder nicht, die ebenfalls mit abgelegt werden. Insgesamt werden 11 Byte zur Beschreibung einer File im Directory abgelegt. Die einzelnen Dateien werden also nacheinander linear auf der Diskette abgelegt. Wird nun eine Datei gelöscht, so bleibt ein Loch auf der Diskette frei. Es muß dann ein sogenannter PACKING-Prozeß eingeleitet werden, bei dem alle nachfolgenden Dateien verschoben werden. Dadurch wird auch das Loch an das Ende der Diskette verschoben.

Anders arbeitet das CP/M-Betriebssystem. Es unterteilt die Diskette in 1024-Byte-Blöcke. Jeder dieser Blöcke wird ein Zahlenwert zugeordnet, der später vom Betriebssystem wieder in Sektor und Track umgerechnet wird. Ein Datenblock kann auch hier wieder einen Namen bekommen, der diesmal 8 Zeichen lang sein darf. Mit Punkt getrennt können nochmals 3 Zeichen eingegeben werden, die die Art des Datenblocks einteilen helfen. So werden z.B. mit .BAS Dateien bezeichnet, die ein BASIC-Programm enthalten, und mit .SRC bzw. .ASM Dateien, die ein Assembler-Sourceprogramm enthalten. .COM bezeichnet Dateien, die ein direkt ausführbares Programm enthalten. Dabei ist es möglich, Programme die mit .COM enden, einfach durch Angabe des Namens, der vor dem Programm steht, zu laden und zu starten.

Hier wird für den Eintrag der Datei unterschiedlich viel Platz im Directory beansprucht. Der minimale Platz ist 32 Byte. Dabei kann eine Datei mit maximal 16 KByte beschrieben werden. Für jede weiteren 16 KByte wird ein neuer Eintrag in der Directory beansprucht. *Abb. 3.1.1-2* zeigt ein Beispiel.

```
0100 00 4D 4F 56 43 50 4D 20 20 43 4F 4D 00 00 00 46  .MOVCPM  COM...F
0110 02 03 04 05 06 07 08 09 0A 00 00 00 00 00 00 00  ................
0120 00 50 49 50 20 20 20 20 20 43 4F 4D 00 00 00 37  .PIP     COM...7
0130 0B 0C 0D 0E 0F 10 11 00 00 00 00 00 00 00 00 00  ................
0140 00 53 55 42 4D 49 54 20 20 43 4F 4D 00 00 00 0A  .SUBMIT  COM.....
0150 12 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0160 00 45 44 20 20 20 20 20 20 43 4F 4D 00 00 00 30  .ED      COM...0
0170 14 15 16 17 18 19 00 00 00 00 00 00 00 00 00 00  ................
0180 00 44 49 53 55 54 4C 20 20 41 53 4D 00 00 54 00  .DISUTL  ASM..T.
0190 75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  u...............
01A0 00 38 30 38 30 20 20 20 20 42 41 4B 00 6F 00 17  .8080    BAK.o..
01B0 83 84 85 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Abb. 3.1.1-2 Beispiel eines CP/M Direktorys

Der Vorteil dieser Art der Datenverwaltung liegt darin, daß ein Packing unnötig wird, da Dateien auch verstreut über die Fläche der Diskette angeordnet sein können. Ferner können mehrere Dateien gleichzeitig zum Schreiben angemeldet werden. Dies ist z.B. bei FDOS nicht möglich. Wird nämlich bei FDOS eine Datei eröffnet, so kann die nächste nicht eröffnet werden, da normalerweise das Ende der ersten Datei noch nicht abzusehen ist. Die nächste Datei muß bekanntlich an die erste anschließen.

Eine Ausnahme ergibt sich, wenn der ersten Datei vonvornherein ein fester großer Platz zugewiesen wird. Bei CP/M tritt dieses Problem nicht auf. Der zweiten Datei wird zunächst der nächste Block mit 1 KByte zugewiesen. Wird dann wieder auf der ersten Datei geschrieben und überschreitet sie den ersten Block, so wird ihr der dritte Block zugewiesen. Dabei entstehen Dateien, die stark ineinander verschachtelt sind. Durch einen Kopier-Prozeß auf eine andere Diskette können sie entwirrt werden. Der einzige Nachteil bei dieser Methode ist, daß schon bei der kleinsten Datei mindestens 1 KByte belegt werden, anstatt nur 128 Byte. Dadurch werden die Disketten schneller voll als mit FDOS.

Es ist auch ein System denkbar, daß jedem Sektor eine Zahl zuordnet, und dadurch der Platz auf der Diskette besser ausgenutzt wird. Das ist z.B. im Betriebssystem ISIS II von INTEL durchgeführt. Der Aufwand steigt allerdings nicht unerheblich.

CP/M ist in mehrere Programme eingeteilt. Es soll zunächst das Programm BIOS besprochen werden. Es kann vom Benutzer geschrieben werden. Es hat die Aufgabe, das Softwarepacket an die vorhandene Hardware anzupassen. Es enthält daher Routinen für Ein- und Ausgabe der Konsole für Ausgabe auf Drucker und für Lesen und Schreiben eines Sektors, sowie Setzen der Adresse von Track und DMA.

Das nächste Programm heißt BDOS. Dies ist das eigentliche Betriebssystem des CP/M. Es greift ausschließlich auf die im BIOS definierten und über eine Sprungtabelle erreichbaren Routinen dort zu.

CCP bezeichnet den CONSOLE COMMAND PROZESSOR. Er hat die Aufgabe, Befehle des Benutzers wie DIR (Directory) auszugeben, also Verzeichnis aller Dateien auszudrucken.

TPA bezeichnet ein Speichergebiet, beginnend mit der Adresse 1ØØH, in das sowohl Benutzerprogramme als auch Dienstprogramme geladen werden. Der CCP beginnt erst von einer höheren Adresse an, je nach maximalem Speicher für den Benutzer. Bei einem Speicher von 16 KByte startet CCP z.B. bei 29ØØH, BDOS bei 32ØØH und BIOS bei 3EØØH.

Dem Benutzer stehen nun zahlreiche Routinen als dem BDOS zur Verfügung. Er kann sie über die Einsprungstelle 5 erreichen. Damit das CP/M-System weiß, welche Dienstleistung verlangt wird, muß dem Befehl CALL auf Adresse 5 ein Parameter in einem Registerpaar übermittelt werden. Im Register DE wird eine Buffer-Adresse und im Register C der Befehl übergeben. Status und Bytes für die Information werden je nach Befehl im Akkumulator übertragen.

Folgende Tabelle gibt die Nummer und Bedeutung der Befehle an:

Nummer	Bedeutung
0	System rücksetzen
1	Konsole lesen
2	Konsole schreiben
3	Reader lesen
4	Punch schreiben
5	Drucker schreiben
6	–
7	IO Status Byte holen
8	IO Status Byte setzen
9	Buffer ausdrucken bis $
10	Buffer füllen von Konsole
11	Zeichen von Konsole da?
12	Lese-/Schreibkopf abheben vom aktuellen Laufwerk
13	BDOS initialisieren und Laufwerk A selektieren, Buffer auf 8ØH voreinstellen
14	Laufwerk X selektieren
15	Datei eröffnen
16	Datei schließen
17	Suchen nach Datei
18	Suchen nach nächster Datei
19	Datei löschen
20	Nächsten Rekord lesen (128 Byte)
21	Nächsten Rekord schreiben
22	Datei kreieren
23	Datei umbenennen
24	Aktuelle Laufwerke
25	Laufwerknummer für nächsten Zugriff
26	DMA-Adresse setzen. 128 Byte Buffer
27	Adresse des Disk-Allocation-Bereiches (Vom STATUS-Kommando verwendet)

3.1.2 Relocator

Abb. 3.1.2-1 zeigt das Listing des Relocator-Programms. Mit .INSERT A:IOPACK wird das schon aus vorhergehenden Abschnitten bekannte IO-Packet eingefügt. Es enthält am Anfang auch eine Sprungtabelle, die zur Anpassung an das Benutzersystem dient.

Als nächstes wird mit INSERT B:CPMINT das Disk Utility Packet, geladen. In diesem Programm werden zunächst einige Konstanten mit Werten belegt, die für die Kommunikation mit dem BDOS-System bestimmt sind. Die erste Routine, die benötigt wird, heißt SETUP. Damit wird eine Datei eröffnet, d.h. für weitere Schreib- und Lesebefehle zugänglich gemacht. Dazu ist der Name in einem Buffer enthalten. Dieser Name wird von dem CP/M-Betriebssystem in der richtigen Form bereits bei dem Aufruf „RELOC name" belegt. Dabei stellt RELOC eine Datei mit der Erweiterung .COM dar, also das Programm in direkt ausführbarer Form. Der Name könnte aber auch per Programm in die im CP/M-Handbuch beschriebenen Speicherzellen geladen werden. (13)

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 1
.MAIN. -


                        .PHEX
                        .PAES
0100                    .LOC 100H
                        ;
                        ;*****************************************
                        ;* RELOCATOR FUER TDL .REL FILES         *
                        ;* U790604                               *
                        ;* BY ROLF-D.KLEIN            V1.0       *
                        ;*****************************************
                        ;
                        ;
                        .INSERT A:TOPACK
                       @;*********************************
                       @;* INCLUDE FILE
                       @;* MIT .INSERT TOPACK
                       @;* RDK 790519
                       @;* BEINHALTET STANDART IO + HEX--
                       @;* UMRECHNUNG
                       @;*********************************
                       @
0100    C3 0108        @JMP START          ;START DES HAUPTPROGRAMMES
0103    C3 F003        @CI:.IMP 0F003H
0106    C3 F009        @CO:.IMP 0F009H
0109    C3 F012        @CSTS: .IMP 0F012H
010C    C3 F01E        @EXIT: .IMP 0F01EH
                       @;
                       @; ROUTINEN EXPF. PRAC-PRHL
                       @; SOWIE CRLF UND PRINT
                       @;
010F                   @PRAC:              ;GIBT A IN ZWEI DIGITS AUS
010F    F5             @PUSH PSW
0110    1F             @RAR
0111    1F             @RAR
0112    1F             @RAR
0113    1F             @RAR
0114    CD 0118        @CALL OUTH
0117    F1             @POP PSW
0118                   @OUTH:
0118    E60F           @ANI 0FH
011A    C630           @ADI "0"
011C    FE3A           @CPI "9"+1
011E    DA 0123        @JC OUTCH
0121    C607           @ADI "A"-"9"-1
0123                   @OUTCH:
0123    4F             @MOV C,A
0124    C3 0106        @JMP CO
                       @;
                       @;
0127                   @PRHL:              ;GIBT HL IN 2 BYTES (4DIGITS )AUS
0127    7C             @MOV A,H
0128    CD 010F        @CALL PRAC
012B    7D             @MOV A,L
012C    18E1           @JMPR PRAC
                       @;
                       @;
```

Abb. 3.1.2-1 Listing des Relocator Programms

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 2
.MAIN. - RELOCATOR V1.0 790604 RDK


                        @; EXPR HOLT ZEICHEN VON DER CONSOLE
                        @; UND SPEICHERT DAS ERGEBNIS IN HL
                        @; DIE EINGABE IST FORMATFREI
                        @; DAS TERMINALZEICHEN WIRD IN A UEBERGEBEN
                        @;
012E                    @EXPR:
012E    21 0000         @LXI H,0            ;ANFANGSWERT
0131                    @EX0:                                          zu Abb. 3.1.2-1
0131    CD 0103         @CALL CI
0134    CD 0123         @CALL DUTCH
0137                    @EX1:
0137    CD 0145         @CALL NIBBLE
013A    DA 0155         @JC EX2
013D    29              @DAD H
013E    29              @DAD H
013F    29              @DAD H
0140    29              @DAD H
0141    B5              @ORA L
0142    6F              @MOV L,A
0143    18EC            @JMPR EX0
0145                    @NIBBLE:
0145    D630            @SUI "0"
0147    D8              @RC
0148    FE17            @CPI "G"-"0"
014A    3F              @CMC
014B    D8              @RC
014C    FE0A            @CPI 10
014E    3F              @CMC
014F    D0              @RNC
0150    D607            @SUI "A"-"9"-1
0152    FE0A            @CPI 10
0154    C9              @RET
0155                    @EX2:
0155    79              @MOV A,C            ;TERMINATOR
0156    FE0D            @CPI 0DH
0158    CA 015C         @JZ ECHO
015B    C9              @RET
                        @;
015C                    @ECHO:
015C    F5              @PUSH PSW
015D    0E0A            @MVI C,0AH
015F    CD 0106         @CALL CO
0162    F1              @POP PSW
0163    4F              @MOV C,A
0164    C9              @RET
                        @;
                        @:
0165                    @CRLF:              ;GIBT CRLF AUF DER CONSOLE AUS
0165    0E0D            @MVI C,0DH
0167    CD 0106         @CALL CO
016A    0E0A            @MVI C,0AH
016C    1898            @JMPR CO
                        @:
                        @;
016E                    @LPRINT:            ;DRUCKT EINEN TEXT BIS 0 AUS
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 3
.MAIN. -- RELOCATOR V1.0 790604 RDK


                        @                       ;ADRESSE IN HL
  016E    7E            @MOV A,M
  016F    23            @INX H
  0170    B7            @ORA A
  0171    C8            @RZ
  0172    CD 0123       @CALL OUTCH
  0175    18F7          @JMPR LPRINT
                        @.DEFINE PRINTCA,%B]=
                        @[                                      zu Abb. 3.1.2-1
                        @LXI H,.+8
                        @CALL LPRINT
                        @JMPR %B
                        @.ASCIZ A
                        @%B:
                        @]
                        @;
                        @; ******** ENDE IOPACK 790519 *********
                         .INSERT B:CPMINT
                        @;****************************************
                        @;* INCLUDE FILE                        *
                        @;* CPM INTERFACE ROUTINEN              *
                        @;* FUER DISK VERARBEITUNG              *
                        @;* RDK 790604                          *
                        @;****************************************
                        @;
  0005                  @BDOS=5               ;DOS EINGANG
  000F                  @OPENF=15             ;FILE OPEN
  0014                  @READF=20             ;READ FUNCTION
  0002                  @TYPEF=2              ;TYPE FUNCTION
  0001                  @CONS=1               ;READ CONSOLE
  000B                  @BRKF=11              ;BREAK KEY (TRUE IF CHAR READY)
  0000                  @RESET=0              ;SYSTEM RESET
  0001                  @READC=1              ;CONSOLE READ
  0003                  @RPUN=3               ;READ PUNCH
  0004                  @WPUN=4               ;WRITE PUNCH
  0005                  @WLIST=5              ;WRITE LIST
  0007                  @IROG=7               ;INTERORG IO STATUS
  0008                  @ALSTAT=8             ;ALTER IO STATUS
  0009                  @BUWRIT=9             ;PRINT CONSOL BUFFER
  000A                  @READCOBU=10          ;READ CONSOL BUFFER
  000C                  @LIFTDI=12            ;LIFT DISK HEAD
  000D                  @RESETSY=13           ;RESET DISK SYSTEM
  000E                  @SELDIS=14            ;SELEKT DISK
  0010                  @FILCLOS=16           ;CLOSE FILE
  0011                  @SEARFIR=17           ;SEARCH FIRST
  0012                  @SEARNX=18            ;SEARCH NEXT
  0013                  @DELFIL=19            ;DELETE FILE
  0014                  @REARECO=20           ;READ RECORD
  0015                  @WRRECO=21            ;WRITE RECORD
  0016                  @CREAFI=22            ;CREATE FILE
  0017                  @RENFIL=23            ;RENAME FILE
  0018                  @INTLOG=24            ;INTEROG LOGIN
  0019                  @INTDIS=25            ;INTEROG DISK
  001A                  @SETDMA=26            ;SET DMA ADR
  001B                  @INTALL=27            ;INTEROG ALLOCATION
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. - RELOCATOR V1.0 790604 RDK


                        @;
005C                    @FCB=5CH            ;FILE CONTROL BLOCK ADR
0080                    @BUFF=80H           ;INPUT DISK BUFFER ADDRESS
                        @;
                        @;
005C                    @FCBDN=FCB+0        ;DISK NAME
005D                    @FCBFN=FCB+1        ;FILE NAME
0065                    @FCBFT=FCB+9        ;DISK FILE TYPE (3CHAR)
0068                    @FCBRL=FCB+12       ;FILE'S CUR REEL NR
006B                    @FCBRC=FCB+15       ;FILE REC COUNT (0..127)
007C                    @FCBCR=FCB+32       ;CUR NEXT REC NR (0..127)
007D                    @FCBLN=FCB+33       ;FCB LENGTH
                        @:
                        @:
0177                    @SETUP:             ;SET UP FILE
0177    11 005C         @LXI D,FCB
017A    0E0F            @MVI C,OPENF
017C    CD 0005         @CALL BDOS          ;FALLS 0FFH IN A DANN FEHLER
017F    FEFF            @CPI 255
0181    37              @STC
0182    C8              @RZ                 ;FEHLER
0183    3F              @CMC
0184    AF              @XRA A
0185    32 007C         @STA FCBCR          ;RECORD 0
0188    C9              @RET
                        @;
0189                    @DISKR:             ;DISK READ
0189    11 005C         @LXI D,FCB
018C    0E14            @MVI C,READF
018E    CD 0005         @CALL BDOS                          zu Abb. 3.1.2-1
0191    37              @STC
0192    3F              @CMC
0193    FE00            @CPI 0
0195    C8              @RZ                 ;KEIN FEHLER
0196    FE01            @CPI 1              ;ENDE FILE
0198    C8              @RZ
0199    37              @STC
019A    C9              @RET                ;FEHLER
                        @;
                         ; UNTERPROGRAMME
                         :
019B                     RIX:
019B    E5               PUSH H
019C    D5               PUSH D
019D    C5               PUSH B             ;ALLES RETTEN
019E    D9               EXX
019F    E5               PUSH H
01A0    D5               PUSH D
01A1    C5               PUSH B
01A2                     GNB:
01A2    3A 01C7          LDA IBP            ;POINTER
01A5    FE80             CPI 80H
01A7    C2 01B1          JNZ G0
                         :
01AA    CD 0189          CALL DISKR
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                                PAGE 5
.MAIN. - RELOCATOR V1.0 790604 RDK


01AD    DA 0253        JC FEHLER
01B0    AF             XRA A
                       ;
01B1                   G0:
01B1    5F             MOV E,A
01B2    1600           MVI D,0
01B4    3C             INR A
01B5    32 01C7        STA TBP
                       ;
01B8    21 0080        LXI H,BUFF
01BB    19             DAD D
01BC    7E             MOV A,M
01BD    C1             POP B
01BE    D1             POP D
01BF    E1             POP H
01C0    D9             EXX
01C1    C1             POP B
01C2    D1             POP D
01C3    E1             POP H
01C4    37             STC
01C5    3F             CMC
01C6    C9             RET
                       ;
01C7    00             TBP: .BYTE 0
                       ;
                       ;
01C8                   START:
01C8    3E80           MVI A,80H
01CA    32 01C7        STA TBP
01CD    CD 0177        CALL SETUP        ;FILE EROEFFNEN
01D0    DA 0253        JC FEHLER         ;FEHLER SONST
                       PRINT "RELOC ENTER BIAS . RELADR :"[
01D3    21 01DB       +LXI H,.+8
01D6    CD 016E       +CALL LPRINT
01D9    181C          +JMPR ..0001
01DB    52454C4F4320+.ASCIZ "RELOC E\
01E1    454E54455220+\NTER B\
01E7    42494153202E+\IAS . \
01ED    2052454C4144+\RELADR\
01F3    52203A00      +\ :"
                      +]
                       ;
01F7                   READ:
01F7    CD 012E        CALL EXPR         ;IN HL EINEN WERT HOLEN
01FA    D60D           SUI 0DH           ;CR ?
01FC    47             MOV B,A           ;RELOC=0
01FD    4F             MOV C,A
01FE    E5             PUSH H
01FF    D1             POP D             ;BIAS NACH DE
0200    2805           JRZ ..R0          ;BEI CR
0202    CD 012E        CALL EXPR         ;SONST RELOCADR HOLEN
0205    E5             PUSH H
0206    C1             POP B             ;NACH BC
0207                   ..R0:
0207    EB             XCHG              ;IN HL BIAS
```

zu Abb. 3.1.2-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 6
.MAIN. - RELOCATOR V1.0 790604 RDK


0208    D9          EXX             ;HL 'BIAS BC'RELOC
0209    CD 0165     CALL CRLF
Q20C                LOD0:
020C    CD 019B     CALL RIX        ;EIN ZEICHEN HOLEN
020F    D63A        SUI ':'         ;DOCH ABSOLUT ?
0211    47          MOV B,A ;RETTEN
0212    E6FE        ANI 0FEH        ;BIT 0 WEG
0214    20F6        JRNZ LOD0       ;WARTEN
0216    57          MOV D,A         ;ZERO CHECKSUM
0217    CD 02DC     CALL SBYTE      ;FILE LENGTH
021A    5F          MOV E,A
021B    CD 02DC     CALL SBYTE      ;LOAD MSB
021E    F5          PUSH PSW
021F    CD 02DC     CALL SBYTE      ;LOAD LSB
0222    D9          EXX
0223    D1          POP D           ;MSB HOLEN
0224    5F          MOV E,A         ;FULL LOAD ADR
0225    C5          PUSH B
0226    D5          PUSH D
0227    E5          PUSH H
0228    19          DAD D           ;BIAS+LOAD ADR
0229    E3          XTHL            ;RESTORE HL'
022A    DDE1        POP X           ;X=BIAS+LOAD
022C    D9          EXX
022D    E1          POP H           ;HL=LOAD ADR
022E    CD 02DC     CALL SBYTE      ;FILE TYPE HOLEN
0231    3D          DCR A           ;1=REL 0=ABS
0232    78          MOV A,B
0233    C1          POP B           ;BC=RELOC
0234    2003        JRNZ ..A
0236    09          DAD B           ;REL
0237    DD09        DADX B          ;X&HL
0239                ..A:
0239    1C          INR E           ;TEST LENGTH
023A    1D          DCR E           ;0=DONE
023B    2832        JRZ DONE
023D    3D          DCR A
023E    286C        JRZ LODR        ;RELATIVE
0240                ..L1:
0240    CD 02DC     CALL SBYTE      ;NEXT
0243    CD 02EF     CALL STORE      ;STORE IT
0246    20F8        JRNZ ..L1
0248                LOD4:
0248    CD 02DC     CALL SBYTE      ;GET CHECKSUM
024B    28BF        JRZ LOD0        ;OK
024D                ERR3:
024D    DDE5        PUSH X
024F    E1          POP H           ;TRANSFER
0250    CD 0127     CALL PRHL       ;CURRENT LOAD ADR
0253                ERR2:
0253                FEHLER:
                    PRINT "FEHLER AUFGETRETEN "[
0253    21 025B    +LXI H,..+8
0256    CD 016E    +CALL LPRINT
0259    1814       +JMPR ..0002
```

zu Abb. 3.1.2-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21
.MAIN. - RELOCATOR V1.0 790604 RDK


025B    4645484C4552+.ASCIZ "FEHLER \
0261    20005546474S+\AUFGET\
0267    5452455445E+\RETEN "\
026D    2000        +\
                    +]
                     :
026F                 DONE:
026F    7C           MOV A,H          ;WENN NULL NICHT MOD
0270    B5           ORA L
0271    CA 0290      JZ FINAL
0274    EB           XCHG
                     PRINT "START ADRESSE :"[
0275    21 027D     +LXI H,.+8
0278    CD 016E     +CALL LPRINT
027B    1810        +.JMPR ..0003
027D    535441525420+.ASCIZ "START A\
0283    4144524553534+\DRESSE\
0289    45203A00    +\ :"
                    +]
028D    CD 0127      CALL PRHL
0290                 FINAL:
                     PRINT "  START MONITOR "[
0290    21 0298     +LXI H,.+8
0293    CD 016E     +CALL LPRINT
0296    1811        +.JMPR ..0004
0298    20205354415 2+.ASCIZ "  START\
029E    54204D4F4E49+\ MONIT\
02A4    544F522000  +\OR "
                    +]
02A9    C3 F01E      JMP 0F01EH       ;***
                     :
02AC                 LODR:
02AC    2E01         MVI L,1          ;BIT COUNTER
02AE                 ..L1:
02AE    CD 02CC      CALL LODCB       ;GET BIT
02B1    3807         JRC ..L3
02B3                 ..L5:
02B3    CD 02EF      CALL STORE       ;STORE IT
02B6    20F6         JRNZ ..L1
02B8    18BF         .JMPR LOD4       ;TEST CHECKSUM
02BA                 ..L3:
02BA    4F           MOV C,A          ;SAVE LOW BYTE
02BB    CD 02CC      CALL LODCB
02BE    47           MOV B,A          ;HIGH BYTE
02BF    D9           EXX
02C0    C5           PUSH B           ;RELOC
02C1    D9           EXX
02C2    E3           XTHL             ;INTO HL
02C3    09           DAD B            ;RELOCATE
02C4    7D           MOV A,L
02C5    CD 02EF      CALL STORE       ;LOW
02C8    7C           MOV A,H
02C9    E1           POP H            ;RESTORE HL
02CA    18E7         .JMPR ..L5
                     :
```

zu Abb. 3.1.2-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 8
.MAIN. - RELOCATOR V1.0 790604 RDK


02CC                    LODCB:
02CC    2D              DCR L           ;COUNT BITS
02CD    2007            JRNZ ..LC1
02CF    CD 02DC         CALL SBYTE
02D2    1D              DCR E           ;COUNT BYTES
02D3    67              MOV H,A         ;SAVE BITS
02D4    2E08            MVI L,8         ;8BITS / BYTE
02D6                    ..LC1:
02D6    CD 02DC         CALL SBYTE
02D9    CB24            SLAR H          ;NEXT BIT
02DB    C9              RET
                        ;
02DC                    SBYTE:
02DC    C5              PUSH B
02DD    CD 02FC         CALL BIBBLE
02E0    07              RLC
02E1    07              RLC
02E2    07              RLC
02E3    07              RLC
02E4    4F              MOV C,A
02E5    CD 02FC         CALL BIBBLE
02E8    B1              ORA C
02E9    4F              MOV C,A
02EA    82              ADD D
02EB    57              MOV D,A
02EC    79              MOV A,C
02ED    C1              POP B
02EE    C9              RET
                        ;
02EF                    STORE:
02EF    DD7700          MOV 0(X),A
02F2    DDBE00          CMP 0(X)        ;PRUEF LESEN
02F5    C2 024D         JNZ ERR3        ;NEIN
02F8    DD23            INX X
02FA    1D              DCR E
02FB    C9              RET
                        ;
02FC                    BIBBLE:
02FC    CD 019B         CALL RIX
02FF    DA 0253         JC FEHLER       ;FEHLER LOAD ERROR
0302    D630            SUI '0'
0304    D8              RC
0305    FE17            CPI 'G'-'0'
0307    3F              CMC
0308    D8              RC
0309    FE0A            CPI 10                  zu Abb. 3.1.2-1
030B    3F              CMC
030C    D0              RNC
030D    D607            SUI 'A'-'9'-1
030F    FE0A            CPI 0AH
0311    C9              RET
                        ;
                        .END
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 9
.MAIN. - RELOCATOR V1.0 790604 RDK
+++++ SYMBOL TABLE +++++

ALSTAT  0008        BDOS    0005        BRKF    000B        BUFF    0080
BUWRIT  0009        CI      0103        CO      0106        CONS    0001
CREAFI  0016        CRLF    0165        CSTS    0109        DELFIL  0013
DISKR   0189        DONE    026F        ECHO    015C        ERR2    0253
ERR3    024D        EX0     0131        EX1     0137        EX2     0155
EXIT    010C        EXPR    012E        FCB     005C        FCBCR   007C
FCBDN   005C        FCBFN   005D        FCBFT   0065        FCBLN   007D
FCBRC   006B        FCBRL   0068        FEHLER  0253        FILCLO  0010
FINAL   0290        GO      01B1        GNB     01A2        IBP     01C7
INTALL  001B        INTDIS  0019        INTLOG  0018        IPOG    0007
LIFTDI  000C        LOD0    020C        LOD4    0248        LODCB   02CC
LODR    02AC        LPRINT  016F        NIBBLE  0145        OPENF   000F
OUTCH   0123        OUTH    0118        PRAC    010F        PRHL    012Z
READCB  000A        READ    01F7        READC   0001        READF   0014
RENFIL  0017        RERECO  0014        RESET   0000        RESETS  000D
RIBBLE  02FC        RIX     019B        RPUN    0003        SBYTE   02DC
SEARFI  0011        SEARNX  0012        SELDIS  000E        SETDMA  001A
SETUP   0177        START   01C8        STORE   02EF        TYPEF   0002
WLIST   0005        WPUN    0004        WRRECO  0015        .BLNK.  0000:03 X
.DATA.  0000* X     .PROG.  0000' X
```

zu Abb. 3.1.2-1

Das nächste Programm heißt DISKR. Es hat die Aufgabe, einen Rekord von 128 Byte aus der Datei zu lesen. Bei jedem Aufruf wird automatisch der nächste Rekord der Datei gelesen. Durch Setzen der Rekord-Nummer wäre es aber auch denkbar, einen wahlfreien (random) Zugriff auf Dateien durchzuführen, der hier jedoch nicht benötigt wird.

Das wichtigste Unterprogramm heißt RIX. Es hat die Aufgabe, ein Byte aus einer Datei zu lesen. Dieses Programm kann vom Benutzer durch eine einfache Routine ersetzt werden. Sie holt z.B. vom Cassettenrecorder ein Byte, um dann ohne Diskette und CP/M arbeiten zu können. Auch das Arbeiten mit FDOS ist möglich, wenn die Routine durch einen Sprung in den FDOS-Monitor ersetzt wird. Dann wird ein Byte von der Disk gelesen. Der Rest des Programms ist unabhängig von CP/M aufgebaut.

In diesem Fall muß RIX nachsehen, ob der Buffer schon leer ist, d.h. ob der Zeiger IBP am Ende des Buffers steht. RIX muß dann die Routine DISKR aufrufen. Das eigentliche Relocating-Programm beginnt mit der Marke START. Als erstes wird die Zelle IPB mit 8ØH belegt, um zu erreichen, daß beim ersten Leseversuch der Buffer zunächst über RIX gefüllt wird. Dann wird SETUP aufgerufen, um die beim Aufruf des Programms angegebene File zu eröffnen und die Parameter initialisieren zu lassen. Der sogenannte File control block FCB befindet sich dabei auf der Adresse 5CH, die voreingestellt ist. Dort steht der Name der Datei. Durch Aufruf des Makros PRINT wird erreicht, daß der Anfangstext ausgegeben wird, dann beginnt das Programm READ, das die Aufgabe hat, die Datei zu lesen und den Inhalt in die Datei zu laden. Dabei können zwei Werte eingegeben werden. Der erste Wert gibt den BIAS (Verschiebewert) an, der zu der Relocalisierungsadresse addiert wird und zusammen mit dieser die Ladeadresse bestimmt. Der zweite Wert ist die Relocalisierungsadresse, die nur bei einer .REL File wirkt. Dann kann das Programm auf eine beliebige Adresse relocalisiert

werden, wo es lauffähig ist. Dabei ist die File im Format 7-Bit-Code (ASCII) abgelegt, welches lesbar ist. Es können aber auch HEX-Files im INTEL-HEX-Format geladen werden, wobei nur der BIAS wirksam ist. Zur Eingabe der im Buch abgedruckten Programme können diese z.B. per Hand in eine Datei mit dem EDITOR eingetragen werden, Format INTEL oder Relocating. Dann wird mit dem Aufruf „RELOC name" die Datei relocalisiert. Sollte ein Prüfsummenfehler vorkommen, so wird die letzte Ladeadresse ausgegeben. Ein Tippfehler kann dann durch erneutes Aufrufen des EDITORs beseitigt werden.

Ist ein Disk nicht vorhanden, so ergibt sich eine andere Möglichkeit. Das Programm kann im Relocating Format auf einer Cassette gespeichert werden, das ebenfalls über einen EDITOR, z.B. TINY EDITOR, zusammen mit der Änderung der Routine RIX eingegeben wurde. Oder RIX wird gleich auf die Konsole geschaltet und das Programm direkt eingegeben. Bei einem Tippfehler erscheint dann eine Fehlermeldung. Es genügt in diesem Fall, den Relocator auf der Adresse 1ØØH neu zu starten und dann die letzte eingegebene Zeile neu einzugeben. Es ist so nicht erforderlich, das ganze Programm neu einzugeben.

Eine File wird mit: ØØØØØØØØØØ beendet. Dabei kann statt der vier unterstrichenen Nullen auch eine Startadresse mit ausgegeben werden. Dann wird diese vom Relocator ebenfalls eingegeben.

3.2 LIFE

An dieser Stelle soll das schon im Buch Mikrocomputersysteme (1) nur dargestellte mathematische Spiel „LIFE" beschrieben werden. Bei LIFE handelt es sich um einen Satz Regeln, der Lebensvorgänge simulieren soll. Dazu wird auf einem Bildschirm, der direkt adressierbar sein soll, der Lebensraum abgebildet.

Es gelten dabei folgende Regeln:

1. Es überlebt derjenige, der auf acht Nachbarfeldern zwei oder drei Nachbarn hat.
2. Es stirbt derjenige, der vier oder mehr Nachbarn hat, oder der nur einen oder keinen Nachbarn hat.
3. Jedes leere Feld mit genau drei Nachbarn wird neu besetzt.

Eine neue Generation wird gleichzeitig für das gesamte in Felder eingeteilte Spielfeld ermittelt.

Abb. 3.2-1 zeigt das Programm LIFE, eine modifizierte Version aus der Zeitschrift DR. DOBBs JOURNAL (14). Das Programm ist für den SOKRATES (SORCERER COMPUTER von EXIDY) ausgelegt. Der Bildwiederholspeicher befindet sich dabei auf der Adresse FØ8ØH. Am Anfang des Programms stehen Sprünge auf die Routine CI, die ein Zeichen im Register A abliefern, und auf die Routine CO. CO liefert über das Register C ein Zeichen ab. Ferner ist ein Sprung auf CSTS vorgesehen. Hier wird der Konsol-Status geliefert, und zwar ØFFH, falls ein Zeichen wartet, sonst Ø.

Der Bildspeicher wird zunächst im RAM aufbereitet. Dies erfolgt auf der Adresse 1ØØØH. Soll er dann angezeigt werden, so wird er mit einer Verschiebe-Routine nach FØ8ØH verschoben und zur Anzeige gebracht. Der Bildspeicher hat hier eine Zeichenmenge von 64 x 24 Zeichen. Zur Anpassung an ein Benutzersystem müssen die Parameter für Zeilenbreite und Anzahl verändert werden.

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 1
.MAIN. -


                        .PABS
                        .PHEX
0100                    .LOC 100H
                        ;
                        ;
                        ;************************************
                        ;* L I F E   V 1 . 0   790106       *
                        ;* RDK REP DR.DOBB N24P11V3         *
                        ;* MODIFIKATION DES OZNAKI LIFE     *
                        ;* BY HARVEY A. COHEN               *
                        ;************************************
                        ;
                        ;
                        ;
0100    C3 0112         JMP START
0103    C3 E009         CI: JMP 0E999H
0106    B7              ORA A
0107    CA 0103         JZ CI
010A    C9              RET
010B    79              CO:MOV A,C      ;CONSOL AUSGABE
010C    C3 E00C         JMP 0E00CH
                        ;
010F    C3 E015         CSTS:JMP 0E015H ;CONSOL STATUS
                        ;
                        ;
                        ; **  PARAMETER  **
                        ;
1000                    VDM=01000H      ;ADRESSE DES DISPLAYS
                                        ;VERSCHOBEN IN RAM
0010                    VDMU=010H
00AA                    ALIFEC=0AAH
007F                    DEADC=7FH
                        ;
                        ;
0112                    START:
                        ;
0112                    CLEAR:
0112    21 1800         LXI H,VDM+0800H         ;LOESCHEN VORBEREITEN
0115    3E0E            MVI A,VDMU-2
0117                    BLANK:
0117    367F            MVI M,DEADC
0119    2B              DCX H
011A    BC              CMP H
011B    C2 0117         JNZ BLANK               ;BIS ALLES GELOESCHT
011E                    READY:
011E    E5              PUSH H
011F    01 FFC0         LXI B,0FFC0H            ;ADRESS INCR B CURSORZEICHEN
0122    11 0040         LXI D,40H               ;SUED ADR INCR
0125                    READIN:
0125    4E              MOV C,M
0126    7E              MOV A,M
0127    70              MOV M,B
0128    CD 01B7         CALL DELAY              ;BLOCK MOVE
012B    CD 0103         CALL CI
012E    71              MOV M,C
```

Abb. 3.2-1 Listing des Programms LIFE

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 2
.MAIN. - LIFE V1.0 ON SOKRATES


012F    0EC0          MVI C,0C0H
0131    FE4E          CPI "N"
0133    CC 01A9       CZ NORTH
0136    FE53          CPI "S"
0138    CC 01AB       CZ SOUTH
013B    FE45          CPI "E"
013D    CC 01AD       CZ EAST
0140    FE57          CPI "W"
0142    CC 01AF       CZ WEST
0145    FE4B          CPI "K"
0147    CC 01B1       CZ KILL
014A    FE4C          CPI "L"
014C    CC 01B4       CZ LIVE
014F    FE43          CPI "C"
0151    CA 0112       JZ CLEAR
0154    FE47          CPI "G"
0156    C2 0125       JNZ READIN
0159    E1            POP H
                      ;
015A                  LIFE:
015A                  NO:
015A    AF            XRA A
015B    86            ADD M
015C    09            DAD B
015D    86            ADD M
015E    23            INX H
015F    86            ADD M
0160    23            INX H
0161    86            ADD M
0162    19            DAD D
0163    86            ADD M
0164    19            DAD D
0165    86            ADD M
0166    2B            DCX H
0167    86            ADD M
0168    2B            DCX H
0169    86            ADD M
016A    23            INX H
016B    09            DAD B
016C    E607          ANI 7
016E    FE06          CPI 6
0170    C4 019A       CNZ MARK
0173    7C            MOV A,H
0174    E618          ANI 18H
0176    FE18          CPI 18H
0178    C2 015A       JNZ NO
017B    CD 01B7       CALL DELAY
017E                  C1:
017E    2B            DCX H
017F    7E            MOV A,M
0180    B7            ORA A
0181    EC 01B4       CPE LIVE
0184    E4 01B1       CPO KILL
0187    7C            MOV A,H
0188    FE0F          CPI VDMU-1
```

zu Abb. 3.2-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 3
.MAIN. - LIFE V1.0 ON SOKRATES


018A    C2 017E       JNZ C1
018D    CD 01B7       CALL DELAY
0190    CD 010F       CALL CSTS
0193    B7            ORA A                    zu Abb. 3.2-1
0194    C2 011E       JNZ READY
0197    C3 015A       JMP LIFE
019A                  MARK:
019A    FE01          CPI 1
019C    CA 01A4       JZ LMARK
019F                  DMARK:
019F    3E7F          MVI A,DEADC
01A1    A6            ANA M
01A2    77            MOV M,A
01A3    C9            RET
0080                  LIFEMARK=80H
01A4                  LMARK:
01A4    3E80          MVI A,LIFEMARK
01A6    B6            ORA M
01A7    77            MOV M,A
01A8    C9            RET
01A9                  NORTH:
01A9    09            DAD B
01AA    C9            RET
01AB                  SOUTH:
01AB    19            DAD D
01AC    C9            RET
01AD                  EAST:
01AD    23            INX H
01AE    C9            RET
01AF                  WEST:
01AF    2B            DCX H
01B0    C9            RET
01B1                  KILL:
01B1    367F          MVI M,DEADC
01B3    C9            RET
01B4                  LIVE:
01B4    36AA          MVI M,ALIFEC
01B6    C9            RET
                      ;
01B7                  DELAY:
                                       ;TRANSPORTROUTINE
                                       ;ALLE AA WERDEN SICHTBAR
                                       ;ALLES ANDERE IST BLANK
01B7    E5            PUSH H
01B8    D5            PUSH D
01B9    C5            PUSH B
01BA    21 1000       LXI H,1000H      ;ANFANG
01BD    11 F080       LXI D,0F080H     ;ZIEL
01C0    01 077F       LXI B,077FH      ;LAENGE
01C3                  LOOPD:
01C3    7E            MOV A,M
01C4    23            INX H
01C5    FEAA          CPI 0AAH         ;ZEICHEN DAS DARGESTELLT WIRD
01C7    CA 01D6       JZ CONT
01CA    FEFF          CPI 0FFH         ;CURSOR
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. - LIFE V1.0 ON SOKRATES


01CC    C2 01D4       JNZ CONT2
01CF    3E2E          MVI A,'.'       ;PUNKT ALS CURSOR
01D1    C3 01D6       JMP CONT
01D4                  CONT2:
01D4    3E20          MVI A,20H       ;BLANK SONST
01D6                  CONT:
01D6    12            STAX D
01D7    13            INX D
01D8    0B            DCX B           ;LAENGE
01D9    78            MOV A,B
01DA    B1            ORA C
01DB    C2 01C3       JNZ LOOPD
01DE    C1            POP B
01DF    D1            POP D
01E0    E1            POP H
01E1    C9            RET             ;OK FERTIG
                      ;
                      ;
                      ;
                      ;                                   zu Abb. 3.2-1
                      .END
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 5
.MAIN. - LIFE V1.0 ON SOKRATES
+++++ SYMBOL TABLE +++++

ALIFEC 00AA      BLANK  0117        C1     017E        CI     0103
CLEAR  0112      CO     010B        CONT   01D6        CONT2  01D4
CSTS   010F      DEADC  007F        DELAY  01B7        DMARK  019F
EAST   01AD      KILL   01B1        LIFE   015A        LIFEMA 0000
LIVE   01B4      LMARK  01A4        LOOPD  01C3        MARK   019A
NO     015A      NORTH  01A9        READIN 0125        READY  011E
SOUTH  01AB      START  0112        VDM    1000        VDMU   0010
WEST   01AF      .BLNK. 0000:03 X   .DATA. 0000*   X    .PROG. 0000'   X
```

Auf der Adresse 112H wird HL mit dem Ende des Bildwiederholspeichers geladen. VDMU gibt die höherwertige Adresse des Bufferspeichers vom RAM an. Auf die Adresse 11FH wird das Zweierkomplement der Anzahl der Zeichen pro Zeile geladen und in die nächste Zeile der Wert selbst. Das Ende des Speichers wird auf der Adresse 174H mit ANI xx CPI xx abgeprüft.

In der Adresse 1BAH wird der Blocktransfer in den Bildwiederholspeicher vorbereitet. Dazu wird das Registerpaar HL mit der Anfangsadresse geladen und DE mit der Zieladresse. BC gibt die Länge des zu übertragenden Blocks an.

Während des Transfers wird eine Umcodierung der Zeichen vorgenommen, so daß es für den Benutzer leicht ist, die interne Darstellung von „Lebenden" in eine sichtbare Darstellung zu verwandeln. AAH ist dabei das Zeichen für „Leben". ØFFH stellt den Cursor dar, der in x „." umcodiert wird. Alle anderen Zeichen werden als Blank (2ØH) ausgegeben.

Eine Figur, die jeder auf dem Bildschirm darzustellen versuchen sollte, ist der sogenannte EATER nach *Abb. 3.2-2.*

```
  x     x
xx  xxxx  xx
  x     x
```

Abb. 3.2-2

Es handelt sich um einen periodischen Oszillator, der nach einigen Generationen wieder in seinen ursprünglichen Zustand zurückkehrt. Der Name EATER kommt daher, daß er die Fähigkeit hat, sogenannte GLIDER zu verschlingen, um dann wieder in den alten Zustand zurückzukehren. Ein dabei entstehender GLIDER ist in *Abb. 3.2-3* dargestellt.

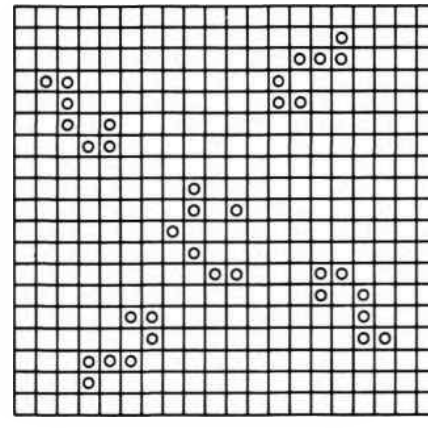
```
 x
x
xxx
```

Abb. 3.2-3

Der GLIDER bewegt sich diagonal über die Bildfläche. Er muß, um den gewünschten Effekt zu erzielen, im richtigen Zeitpunkt an die richtige Stelle vom EATER treffen.

Es folgen einige Abbildungen, die weitere Beispiele zeigen. Sie sind der Zeitschrift BYTE vom Dezember 1978 entnommen. *Abb. 3.2-4* zeigt einen sogenannten EATER BOUND OSCILLATOR, der nach 52 Generationen seine ursprüngliche Form wieder erlangt. In *Abb. 3.2-5* sind verschiedene GLIDER dargestellt, die sich in einer Richtung über den Bildschirm bewegen. *Abb. 3.2-6* stellt ein besonders interessantes Objekt dar, einen GLIDER GUN. Dieses System hat die Eigenschaft, GLIDER zu produzieren, die sich dann entfernen und gegebenenfalls mit einem EATER wieder beseitigt werden können. Dieser GLIDER GUN arbeitet in einer Periode von 30 Generationen. *Abb. 3.2-7* zeigt schließlich einen GLIDER GUN mit dem Namen NEWGUN, der 46 Perioden benötigt, um in den Ausgangszustand zurückzukehren.

Abb. 3.2-4 EATER BOUND OSCILLATOR



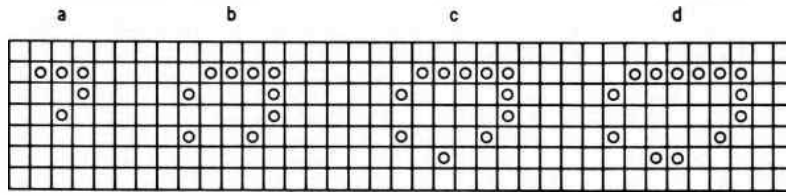


Abb. 3.2-5 Verschiedene GLIDER

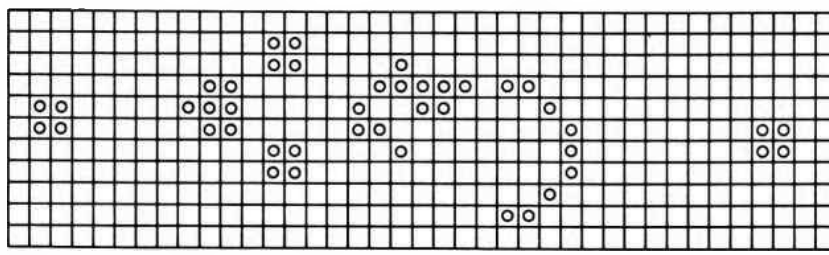


Abb. 3.2.-6 GLIDER GUN mit 30 Generationen

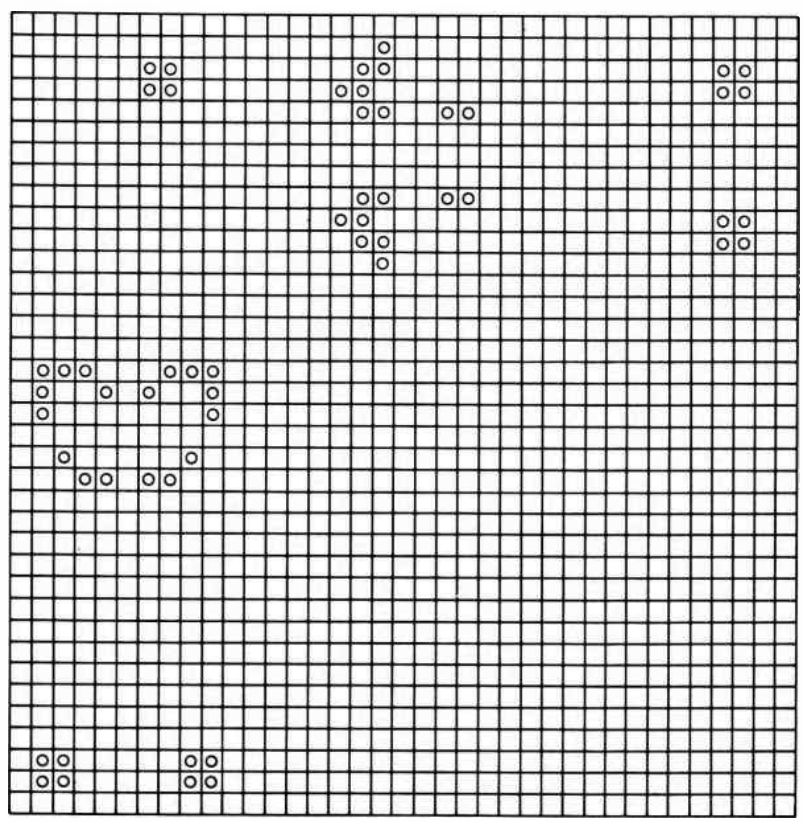


Abb. 3.2-7 GLIDER GUN mit 46 Generationen

3.3 Boolescher Formel-Analysator

Es soll hier ein Programm besprochen werden, das in dieser Version in der Lage ist, eine Boolesche Formel mit maximal sechs Variablen zu analysieren und dessen Wahrheitstabelle auszugeben. Die Formel kann an beliebiger Stelle Klammern und Negationen enthalten.

Abb. 3.3-1 zeigt das Programm. Es beginnt wieder mit dem bekannten IOPACK. Bei der Marke START beginnt das eigentliche Programm. Nach dem Starten des Programms wird eine beliebige Formel eingegeben, wobei die Buchstaben A bis F die einzelnen Variablen sind. Die Eingabe geschieht in der Routine EING, und die Formel wird in einem Buffer auf der Adresse BUFA zwischengespeichert. Mit CR wird die Eingabe abgeschlossen. Als nächstes wird die Überschrift ausgegeben. In das Register B wird der Zähler gelegt, der mit 64 vorbesetzt wird. Die sechs Variablen ergeben nämlich 64 Eingangskombinationen.

Dieser Zähler wird dann Bit für Bit ausgegeben und gleichzeitig in die Variablen X6 bis X1 gespeichert, wobei immer nur das erste Bit (Bit 0) relevant ist. Es folgt dann der Aufruf des Unterprogramms EXXPR, welches die Analyse der Booleschen Formel mit den aktuellen Parametern in X6 bis X1 vornimmt. Dabei erhält im Register C Bit 0 den logischen Wert 1 oder 0.

OUTP gibt ein Bit aus und Blank ein Leerzeichen. Dadurch werden die einzelnen Bits im Ausgabeformat getrennt. Den eigentlichen Kern stellt EXXPR dar. Es handelt sich um ein rekursiv programmiertes Unterprogramm. Dazu soll *Abb. 3.3-2* als Erklärung dienen. Hier sind sogenannte Syntax-Diagramme dargestellt, wie sie z.B. zur Definition der Sprache PASCAL verwendet werden. Die Syntax-Diagramme beschreiben eine Sprache. Sie definieren – wie der Name sagt – die Syntax, d.h. die Aufeinanderfolge von Worten, nicht aber die Semantik, also die Bedeutung der Worte und der zusammengesetzten Worte.

Im ersten Diagramm ist die Definition von EXXPR dargestellt. Das ist die Definition eines Booleschen Ausdrucks. Er besteht zunächst aus einem Term, der – falls er von einem ODER-Zeichen gefolgt wird – wieder von einem Term gefolgt wird, bis kein ODER-Zeichen mehr kommt. Ein Term ist definiert als ein Gebilde, bei dem ein Faktor an erster Stelle steht, der – falls er von einem UND-Zeichen gefolgt wird, wieder von einem Faktor gefolgt sein muß, bis kein UND-Zeichen mehr kommt.

Ein Faktor schließlich ist definiert durch eine Konstante oder eine Variable oder ein NICHT-Zeichen, die von einem weiteren Faktor gefolgt werden, oder aber durch ein Klammer-auf-Zeichen, seinerseits gefolgt von einem Booleschen Ausdruck (expression), weiter gefolgt von einem Klammer-zu-Zeichen. Hier wird die rekursive Definitionstechnik erkennbar. Im Faktor kommt einmal wieder Faktor vor, ein anderes Mal sogar EXXPR. Das bedeutet, daß an dieser Stelle wieder zu EXXPR gegangen werden muß, und zwar durch alle Syntax-Diagramme hindurch bis an das Ende von EXXPR, um dann wieder in Faktor an der Stelle vor Klammer-zu anzukommen. Dabei kann in EXXPR wieder EXXPR aufgerufen werden. Variable kann schließlich einer der Buchstaben A bis F sein, oder auch logisch 0 oder 1 (Const 1 oder Const 0).

In dem Programm EXXPR wird als erstes in HL die Anfangsadresse des Buffer geladen, die den Booleschen Ausdruck enthält. EXXPR1 ist der Beginn der eigentlichen rekursiven Routine. Zunächst wird mit CALL TERM ein Term aufgerufen. Der Wert vom Term steht ebenfalls im Register C an BIT 0. Folgt auf Term kein ODER-Zeichen, so wird mit RNZ der Vorgang beendet, und der Wert wird dem Hauptprogramm über-

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                          PAGE 1
↓.MAIN. -

                        ;
                        .PABS
                        .PHEX
0100                    .LOC 100H
                        ;*************************************
                        ;* B O O L   BOOLSCHER FORMELRECHNER *
                        ;* 790602 RDK                        *
                        ;*************************************
                        ;
                        .INSERT A:IOPACK
                       @;*********************************
                       @;* INCLUDE FILE
                       @;* MIT .INSERT IOPACK
                       @;* RDK 790519
                       @;* BEINHALTET STANDART IO + HEX-
                       @;* UMRECHNUNG
                       @;*********************************
                       @
0100    C3 0177        @JMP START        ;START DES HAUPTPROGRAMMES
0103    C3 F003        @CI:JMP 0F003H
0106    C3 F009        @CO:JMP 0F009H
0109    C3 F012        @CSTS: JMP 0F012H
010C    C3 F01E        @EXIT: JMP 0F01EH
                       @;
                       @; ROUTINEN EXPR, PRAC,PRHL
                       @; SOWIE CRLF UND PRINT
                       @;
010F                   @PRAC:            ;GIBT A IN ZWEI DIGITS AUS
010F    F5             @PUSH PSW
0110    1F             @RAR
0111    1F             @RAR
0112    1F             @RAR
0113    1F             @RAR
0114    CD 0118        @CALL OUTH
0117    F1             @POP PSW
0118                   @OUTH:
0118    E60F           @ANI 0FH
011A    C630           @ADI "0"
011C    FE3A           @CPI "9"+1
011E    DA 0123        @JC OUTCH
0121    C607           @ADI "A"-"9"-1
0123                   @OUTCH:
0123    4F             @MOV C,A
0124    C3 0106        @JMP CO
                       @;
                       @;
0127                   @PRHL:            ;GIBT HL IN 2 BYTES (4DIGITS )AUS
0127    7C             @MOV A,H
0128    CD 010F        @CALL PRAC
012B    7D             @MOV A,L
012C    18E1           @JMPR PRAC
                       @;
                       @;
                       @; EXPR HOLT ZEICHEN VON DER CONSOLE
                       @; UND SPEICHERT DAS ERGEBNIS IN HL
```

Abb. 3.3-1 Listing des Boolschen Formel-Analysators

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 2
.MAIN. -


                        @; DIE EINGABE IST FORMATFREI
                        @; DAS TERMINALZEICHEN WIRD IN A UEBERGEBEN
                        @;
012E                    @EXPR:
012E    21 0000         @LXI H,0            ;ANFANGSWERT
0131                    @EX0:
0131    CD 0103         @CALL CI
0134    CD 0123         @CALL OUTCH
0137                    @EX1:
0137    CD 0145         @CALL NIBBLE
013A    DA 0155         @JC EX2
013D    29              @DAD H                                  zu Abb. 3.3-1
013E    29              @DAD H
013F    29              @DAD H
0140    29              @DAD H
0141    B5              @ORA L
0142    6F              @MOV L,A
0143    18EC            @JMPR EX0
0145                    @NIBBLE:
0145    D630            @SUI '0'
0147    D8              @RC
0148    FE17            @CPI 'G'-'0'
014A    3F              @CMC
014B    D8              @RC
014C    FE0A            @CPI 10
014E    3F              @CMC
014F    D0              @RNC
0150    D607            @SUI 'A'-'9'-1
0152    FE0A            @CPI 10
0154    C9              @RET
0155                    @EX2:
0155    79              @MOV A,C            ;TERMINATOR
0156    FE0D            @CPI 0DH
0158    CA 015C         @JZ ECHO
015B    C9              @RET
                        @;
015C                    @ECHO:
015C    F5              @PUSH PSW
015D    0E0A            @MVI C,0AH
015F    CD 0106         @CALL CO
0162    F1              @POP PSW
0163    4F              @MOV C,A
0164    C9              @RET
                        @;
                        @;
0165                    @CRLF:              ;GIBT CRLF AUF DER CONSOLE AUS
0165    0E0D            @MVI C,0DH
0167    CD 0106         @CALL CO
016A    0E0A            @MVI C,0AH
016C    1898            @JMPR CO
                        @;
                        @;
016E                    @LPRINT:            ;DRUCKT EINEN TEXT BIS 0 AUS
                        @                   ;ADRESSE IN HL
016E    7E              @MOV A,M
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 3
.MAIN. -


016F    23          @INX H
0170    B7          @ORA A
0171    C8          @RZ
0172    CD 0123     @CALL OUTCH
0175    18F7        @JMPR LPRINT
                    @.DEFINE PRINTC A,%B C=
                    @C
                    @LXI H,.+8
                    @CALL LPRINT
                    @JMPR %B
                    @.ASCIZ A
                    @%B:                                    zu Abb. 3.3-1
                    @C
                    @;
                    @; ******** ENDE IOPACK 790519 *********
                     ;
0177                 START:
                     PRINT "BOOLSCHER FORMELRECHNER" C0DH C0AH CC
0177    21 017F     +LXI H,.+8
017A    CD 016E     +CALL LPRINT
017D    181A        +JMPR ..0001
017F    424F4F4C5343+.ASCIZ "BOOLSCH\
0185    48455220464F+\ER FOR\
018B    524D454C5245+\MELREC\
0191    43484E45520D+\HNER" C0DH C0AH C\
0197    0A00        +\
                    +C
                     PRINT "FORMEL:"C
0199    21 01A1     +LXI H,.+8
019C    CD 016E     +CALL LPRINT
019F    1808        +JMPR ..0002
01A1    464F524D454C+.ASCIZ "FORMEL:"\
01A7    3A00        +\
                    +C
01A9    21 031D      LXI H,BUFA
01AC                 EING:
01AC    CD 0103      CALL CI
01AF    4F           MOV C,A
01B0    CD 0106      CALL CO
01B3    77           MOV M,A
01B4    23           INX H
01B5    FE0D         CPI 0DH
01B7    C2 01AC      JNZ EING
                     ;
01BA    CD 0165      CALL CRLF
                     PRINT "F E D C B A  X"C0DH C0AH CC
01BD    21 01C5     +LXI H,.+8
01C0    CD 016E     +CALL LPRINT
01C3    1811        +JMPR ..0003
01C5    462045204420+.ASCIZ "F E D C\
01CB    432042204120+\ B A  \
01D1    20580D0A00  +\X"C0DH C0AH C
                    +C
                     ; FORMEL EINGEGEBEN
                     ;
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 4
.MAIN. -


01D6    AF          XRA A
01D7    32 031C     STA COUNT       ;BINAERZAEHLER
                    ;
01DA    0640        MVI B,64        ;16 MAL
01DC                LOOP:
01DC    C5          PUSH B          ;B RETTEN
01DD    3A 031C     LDA COUNT       ;IN VARIABLE
01E0    07          RLC             ;HIGH ORDER ZUERST
01E1    07          RLC
01E2    07          RLC
01E3    CD 0233     CALL OUTP       ;AUSGEBEN
01E6    32 031B     STA X6          ;BIT 0 RELEVANT
01E9    07          RLC
01EA    CD 0233     CALL OUTP
01ED    32 031A     STA X5
01F0    07          RLC
01F1    CD 0233     CALL OUTP
01F4    32 0319     STA X4
01F7    07          RLC
01F8    CD 0233     CALL OUTP
01FB    32 0318     STA X3
01FE    07          RLC
01FF    CD 0233     CALL OUTP
0202    32 0317     STA X2
0205    07          RLC
0206    CD 0233     CALL OUTP
0209    32 0316     STA X1          ;ALLE 6 VARIABLE
020C    CD 022B     CALL BLANK      ;BLANK AUSGEBEN
020F    CD 0255     CALL EXXPR      ;ERGEBNIS IN BC
0212    79          MOV A,C
0213    CD 0233     CALL OUTP       ;DRUCKEN
0216    CD 0165     CALL CRLF       ;VORSCHUB
0219    C1          POP B           ;ZAEHLER
021A    3A 031C     LDA COUNT
021D    3C          INR A
021E    32 031C     STA COUNT
0221    05          DCR B
0222    C2 01DC     JNZ LOOP
0225    CD 0165     CALL CRLF
0228    C3 0177     JMP START
                    ;
022B                BLANK:
022B    F5          PUSH PSW
022C    0E20        MVI C,' '
022E    CD 0106     CALL CO
0231    F1          POP PSW
0232    C9          RET
                    ;
0233                OUTP:
                                    ;IN A BIT 0 STEHT WERT
0233    F5          PUSH PSW        ;RETTEN
0234    E601        ANI 1
0236    CA 0243     JZ ZERO
0239    0E31        MVI C,'1'
023B    CD 0106     CALL CO
```

zu Abb. 3.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                PAGE 5
.MAIN. -


023E    CD 022B         CALL BLANK              zu Abb. 3.3-1
0241    F1              POP PSW
0242    C9              RET
0243                    ZERO:
0243    0E30            MVI C,'0'
0245    CD 0106         CALL CO
0248    CD 022B         CALL BLANK
024B    F1              POP PSW
024C    C9              RET
                        ;
                        ;
024D                    IGBN:
024D    7E              MOV A,M
024E    FE20            CPI ' '
0250    C0              RNZ
0251    23              INX H
0252    C3 024D         JMP IGBN            ;IGNORE BLANKS
                        ;
                        ;
0255                    EXXPR:
0255    21 031D         LXI H,BUFA
0258                    EXXPR1:
0258    CD 026E         CALL TERM
025B                    EXLP:
025B    CD 024D         CALL IGBN
025E    7E              MOV A,M
025F    FE2B            CPI '+'             ;OR=+
0261    C0              RNZ                 ;FERTIG
0262    23              INX H
0263    C5              PUSH B              ;RETTEN BC
0264    CD 026E         CALL TERM
0267    D1              POP D
0268    7B              MOV A,E
0269    B1              ORA C
026A    4F              MOV C,A             ;ODERN
026B    C3 025B         JMP EXLP
                        ;
026E                    TERM:
026E    CD 0284         CALL FACTOR
0271                    TELP:
0271    CD 024D         CALL IGBN
0274    7E              MOV A,M
0275    FE2A            CPI '*'             ;AND=*
0277    C0              RNZ
0278    23              INX H
0279    C5              PUSH B
027A    CD 0284         CALL FACTOR
027D    D1              POP D
027E    7B              MOV A,E
027F    A1              ANA C
0280    4F              MOV C,A             ;UND VERKN.
0281    C3 0271         JMP TELP
                        ;
0284                    FACTOR:
0284    CD 024D         CALL IGBN
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 6
.MAIN. -
```

zu Abb. 3.3-1

```
0287    7E          MOV A,M
0288    FE28        CPI '('
028A    C2 02B2     JNZ FAC1
028D    23          INX H
028E    CD 0258     CALL EXXPR1
0291    CD 024D     CALL IGBN
0294    7E          MOV A,M
0295    FE29        CPI ')'
0297    23          INX H
0298    C8          RZ
                    PRINT 'KLAMMER FEHLT'[0DH] [0AH]
0299    21 02A1     +LXI H,.+8
029C    CD 016E     +CALL LPRINT
029F    1810        +JMPR ..0004
02A1    4B4C414D4D45+.ASCIZ 'KLAMMER\
02A7    5220464548 4C+\ FEHLT'[0DH\
02AD    540D0A00    +\] [0AH]
                    +]
02B1    C9          RET
                    ;
02B2                FAC1:
02B2    CD 024D     CALL IGBN
02B5    7E          MOV A,M
02B6    FE5E        CPI '↑'          ;NOT=↑
02B8    C2 02C3     JNZ FAC2
02BB    23          INX H
02BC    CD 0284     CALL FACTOR
02BF    79          MOV A,C
02C0    2F          CMA
02C1    4F          MOV C,A
02C2    C9          RET
                    ;
02C3                FAC2:
                            ;VARIABLENWERT HOLEN
02C3    CD 024D     CALL IGBN
02C6    7E          MOV A,M ;A B C D E F SIND ERLAUBT
02C7    23          INX H
02C8    FE41        CPI 'A'
02CA    CA 02F2     JZ ASET
02CD    FE42        CPI 'B'
02CF    CA 02F7     JZ BSET
02D2    FE43        CPI 'C'
02D4    CA 02FC     JZ CSET
02D7    FE44        CPI 'D'
02D9    CA 0301     JZ DSET
02DC    FE45        CPI 'E'
02DE    CA 0306     JZ ESET
02E1    FE46        CPI'F'
02E3    CA 030B     JZ FSET
02E6    FE31        CPI '1'
02E8    CA 0310     JZ SET1
02EB    FE30        CPI '0'
02ED    CA 0313     JZ SET0
02F0    2B          DCX H
02F1    C9          RET
```

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 7
.MAIN. -


                        ;
02F2                    ASET:
02F2    3A 0316         LDA X1
02F5    4F              MOV C,A
02F6    C9              RET
                        ;
02F7                    BSET:
02F7    3A 0317         LDA X2
02FA    4F              MOV C,A
02FB    C9              RET
                        ;
02FC                    CSET:
02FC    3A 0318         LDA X3
02FF    4F              MOV C,A
0300    C9              RET
                        ;
0301                    DSET:
0301    3A 0319         LDA X4
0304    4F              MOV C,A
0305    C9              RET
                        ;
0306                    ESET:
0306    3A 031A         LDA X5
0309    4F              MOV C,A
030A    C9              RET
                        ;
030B                    FSET:
030B    3A 031B         LDA X6
030E    4F              MOV C,A
030F    C9              RET
                        ;
                        ;
0310                    SET1:
0310    0E01            MVI C,1
0312    C9              RET
                        ;
0313                    SET0:
0313    0E00            MVI C,0
0315    C9              RET
                        ;
                        ;* SPEICHERZELLEN*

0316    00              X1: .BYTE 0
0317    00              X2: .BYTE 0
0318    00              X3: .BYTE 0
0319    00              X4: .BYTE 0
031A    00              X5: .BYTE 0
031B    00              X6: .BYTE 0
031C    00              COUNT: .BYTE 0
                        ;
031D                    BUFA: .BLKB 40
                        ;
                        .END
```

zu Abb. 3.3-1

```
TDL Z80 CP/M DISK ASSEMBLER VERSION 2.21                    PAGE 8
.MAIN. -
+++++ SYMBOL TABLE +++++

ASET    02F2        BLANK   022B        BSET    02F7        BUFA    031D
CI      0103        CO      0106        COUNT   031C        CRLF    0165
CSET    02FC        CSTS    0109        DSET    0301        ECHO    015C
EING    01AC        ESET    0306        EX0     0131        EX1     0137
EX2     0155        EXIT    010C        EXLP    025B        EXPR    012E
EXXPR   0255        EXXPR1  0258        FAC1    02B2        FAC2    02C3
FACTOR  0284        FSET    030B        IGBN    024D        LOOP    01DC
LPRINT  016E        NIBBLE  0145        OUTCH   0123        OUTH    0118
OUTP    0233        PRAC    010F        PRHL    0127        SET0    0313
SET1    0310        START   0177        TELP    0271        TERM    026E
X1      0316        X2      0317        X3      0318        X4      0319
X5      031A        X6      031B        ZERO    0243        .BLNK.  0000:03 X
.DATA.  0000*   X   .PROG.  0000'   X
```

zu Abb. 3.3-1

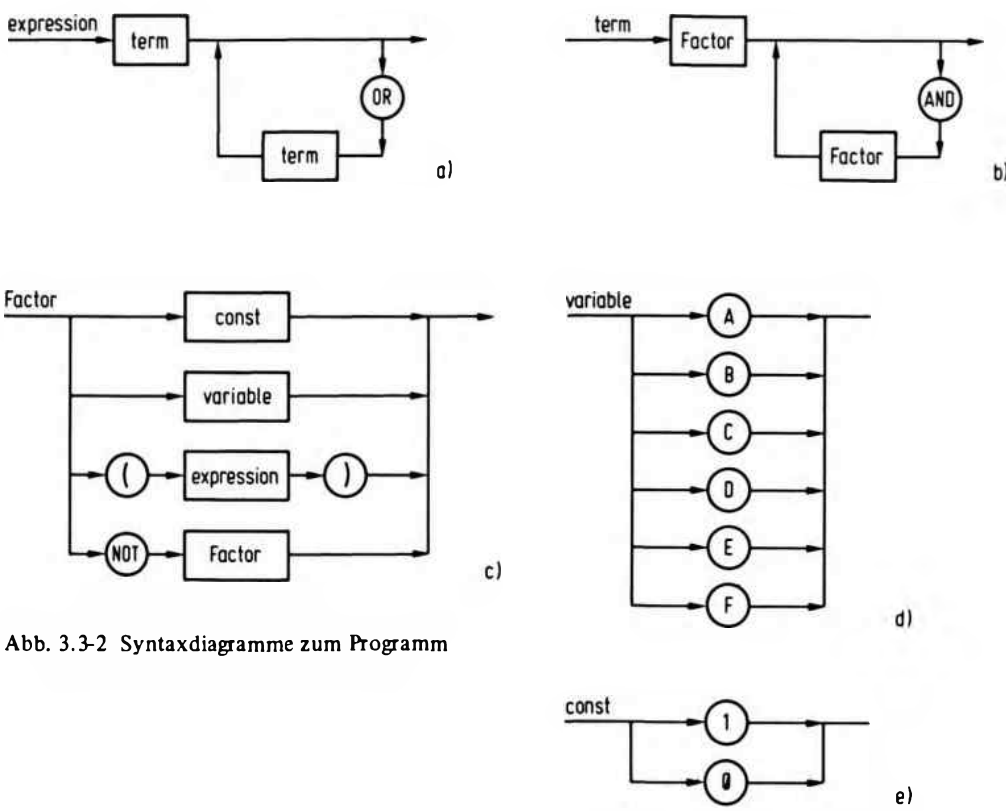


Abb. 3.3-2 Syntaxdiagramme zum Programm

geben. Andernfalls wird der Wert mit PUSH B gerettet und erneut die Routine Term aufgerufen. Dann wird nach einem POP D, welches den Wert in das Register E schafft, und mit MOV A,E ORA C die ODER-Verknüpfung durchgeführt.

Die Routine TERM ist im Prinzip genau so programmiert. Hier wird aber eine UND-Verknüpfung durchgeführt, falls das UND-Zeichen auftritt.

Die Routine IGBN hat in beiden Fällen die Aufgabe, überflüssige Leerzeichen zu ignorieren. Diese Routine hätte im Prinzip auch schon in die Leseroutine eingebunden werden können. Faktor ist etwas komplexer, als die beiden vorhergehenden Routinen EXXPR1 und TERM.

Falls am Anfang eine Klammer-auf auftauchen sollte, so wird mit einem CALL EXXPR1 die Rekursion ausgeführt und anschließend das Klammerzeichen erwartet. Tritt dieses nicht auf, so liegt ein Syntax-Fehler vor, es wird eine Fehlermeldung ausgegeben. Wenn nicht Klammer-auf vorliegt, so könnte ein Negationszeichen vorgekommen sein. Dies wird in FAC1 abgefragt. Handelt es sich um dieses Zeichen, so wird Faktor rekursiv aufgerufen. Nach dem Aufruf wird mit einem Befehl CMA das BIT 0 des Registers C komplementiert, was einer Negation entspricht. War es nicht das NICHT-Zeichen, so kann es nur noch eine Variable gewesen sein. Dann wird auf die Variablen A bis F geprüft und ein Wert der Zellen X1 bis X6 in das Register C geladen. Falls es eine Konstante entsprechend logisch 0 oder logisch 1 war, wird der entsprechende Wert ebenfalls in das Register C geladen. *Abb. 3.3-3* zeigt einige Beispiele mit dem hier vorgestellten Booleschen Formel-Analysator.

```
BOOLSCHER FORMELRECHNER
FORMEL:↑(A+↑B+↑C)+↑A*D)*E*F+C*A

F E D C B A  X
0 0 0 0 0 0  0
0 0 0 0 0 1  0
0 0 0 0 1 0  0
0 0 0 0 1 1  0
0 0 0 1 0 0  0
0 0 0 1 0 1  1
0 0 0 1 1 0  1
0 0 0 1 1 1  1
0 0 1 0 0 0  0
0 0 1 0 0 1  0
0 0 1 0 1 0  0
0 0 1 0 1 1  0
0 0 1 1 0 0  0
0 0 1 1 0 1  1
0 0 1 1 1 0  1
0 0 1 1 1 1  1
0 1 0 0 0 0  0
0 1 0 0 0 1  0
0 1 0 0 1 0  0
0 1 0 0 1 1  0
0 1 0 1 0 0  0
0 1 0 1 0 1  1
0 1 0 1 1 0  1
0 1 0 1 1 1  1
0 1 1 0 0 0  0
0 1 1 0 0 1  0
0 1 1 0 1 0  0
0 1 1 0 1 1  0
0 1 1 1 0 0  0
0 1 1 1 0 1  1
0 1 1 1 1 0  1
0 1 1 1 1 1  1
```

```
BOOLSCHER FORMELRECHNER
FORMEL:A*B+(A*↑C*↑D)+↑E+A*↑F

F E D C B A  X
0 0 0 0 0 0  1
0 0 0 0 0 1  1
0 0 0 0 1 0  1
0 0 0 0 1 1  1
0 0 0 1 0 0  1
0 0 0 1 0 1  1
0 0 0 1 1 0  1
0 0 0 1 1 1  1
0 0 1 0 0 0  1
0 0 1 0 0 1  1
0 0 1 0 1 0  1
0 0 1 0 1 1  1
0 0 1 1 0 0  1
0 0 1 1 0 1  1
0 0 1 1 1 0  1
0 0 1 1 1 1  1
0 1 0 0 0 0  0
0 1 0 0 0 1  1
0 1 0 0 1 0  0
0 1 0 0 1 1  1
0 1 0 1 0 0  0
0 1 0 1 0 1  1
0 1 0 1 1 0  0
0 1 0 1 1 1  1
0 1 1 0 0 0  0
0 1 1 0 0 1  1
0 1 1 0 1 0  0
0 1 1 0 1 1  1
0 1 1 1 0 0  0
0 1 1 1 0 1  1
0 1 1 1 1 0  0
0 1 1 1 1 1  1
```

Abb. 3.3-3

```
BOOLSCHER FORMELRECHNER
FORMEL:↑(A+↑B+↑C)+↑A*D)*E*F+C*A

F E D C B A X
1 0 0 0 0 0 0
1 0 0 0 0 1 0
1 0 0 0 1 0 0
1 0 0 0 1 1 0
1 0 0 1 0 0 0
1 0 0 1 0 1 1
1 0 0 1 1 0 1
1 0 0 1 1 1 1
1 0 1 0 0 0 0
1 0 1 0 0 1 0
1 0 1 0 1 0 0
1 0 1 0 1 1 0
1 0 1 1 0 0 0
1 0 1 1 0 1 1
1 0 1 1 1 0 1
1 0 1 1 1 1 1
1 1 0 0 0 0 0
1 1 0 0 0 1 0
1 1 0 0 1 0 0
1 1 0 0 1 1 0
1 1 0 1 0 0 0
1 1 0 1 0 1 1
1 1 0 1 1 0 1
1 1 0 1 1 1 1
1 1 1 0 0 0 1
1 1 1 0 0 1 0
1 1 1 0 1 0 1
1 1 1 0 1 1 0
1 1 1 1 0 0 1
1 1 1 1 0 1 1
1 1 1 1 1 0 1
1 1 1 1 1 1 1
```

```
BOOLSCHER FORMELRECHNER
FORMEL:A*B+(A*↑C*↑D)+↑E+A*↑F

F E D C B A X
1 0 0 0 0 0 1
1 0 0 0 0 1 1
1 0 0 0 1 0 1
1 0 0 0 1 1 1
1 0 0 1 0 0 1
1 0 0 1 0 1 1
1 0 0 1 1 0 1
1 0 0 1 1 1 1
1 0 1 0 0 0 1
1 0 1 0 0 1 1
1 0 1 0 1 0 1
1 0 1 0 1 1 1
1 0 1 1 0 0 1
1 0 1 1 0 1 1
1 0 1 1 1 0 1
1 0 1 1 1 1 1
1 1 0 0 0 0 0
1 1 0 0 0 1 1
1 1 0 0 1 0 0
1 1 0 0 1 1 1
1 1 0 1 0 0 0
1 1 0 1 0 1 0
1 1 0 1 1 0 0
1 1 0 1 1 1 1
1 1 1 0 0 0 0
1 1 1 0 0 1 0
1 1 1 0 1 0 0
1 1 1 0 1 1 1
1 1 1 1 0 0 0
1 1 1 1 0 1 0
1 1 1 1 1 0 0
1 1 1 1 1 1 1
```

zu Abb. 3.3-3

```
BOOLSCHER FORMELRECHNER FORMEL:A+B+(C*(D+↑E))+↑(↑F+↑A)

F E D C B A X
0 0 0 0 0 0 0
0 0 0 0 0 1 1
0 0 0 0 1 0 1
0 0 0 0 1 1 1
0 0 0 1 0 0 1
0 0 0 1 0 1 1
0 0 0 1 1 0 1
0 0 0 1 1 1 1
0 0 1 0 0 0 0
0 0 1 0 0 1 1
0 0 1 0 1 0 1
0 0 1 0 1 1 1
0 0 1 1 0 0 1
0 0 1 1 0 1 1
0 0 1 1 1 0 1
0 0 1 1 1 1 1
0 1 0 0 0 0 0
0 1 0 0 0 1 1
0 1 0 0 1 0 1
0 1 0 0 1 1 1
0 1 0 1 0 0 0

0 1 0 1 0 1 1
0 1 0 1 1 0 1
0 1 0 1 1 1 1
0 1 1 0 0 0 1
0 1 1 0 0 1 1
0 1 1 0 1 0 1
0 1 1 0 1 1 1
0 1 1 1 0 0 1
0 1 1 1 0 1 1
0 1 1 1 1 0 1
0 1 1 1 1 1 1
1 0 0 0 0 0 0
1 0 0 0 0 1 1
1 0 0 0 1 0 1
1 0 0 0 1 1 1
1 0 0 1 0 0 1
1 0 0 1 0 1 1
1 0 0 1 1 0 1
1 0 0 1 1 1 1
1 0 1 0 0 0 0
1 0 1 0 0 1 1
1 0 1 0 1 0 1

1 0 1 0 1 1 1
1 0 1 1 0 0 1
1 0 1 1 0 1 1
1 0 1 1 1 0 1
1 0 1 1 1 1 1
1 1 0 0 0 0 0
1 1 0 0 0 1 1
1 1 0 0 1 0 1
1 1 0 0 1 1 1
1 1 0 1 0 0 1
1 1 0 1 0 1 1
1 1 0 1 1 0 1
1 1 0 1 1 1 1
1 1 1 0 0 0 0
1 1 1 0 0 1 0
1 1 1 0 1 0 0
1 1 1 0 1 1 1
1 1 1 1 0 0 0
1 1 1 1 0 1 0
1 1 1 1 1 0 0
1 1 1 1 1 1 1
```

zu Abb. 3.3-3

Literaturverzeichnis

[1] Klein, Rolf-Dieter: Mikrocomputersysteme. 2. Auflage. Franzis-Verlag, München.

[2] Klein, Dr.-Ing. Paul E.: Das Oszilloskop. Franzis-Verlag, München.

[3] Klein, Dr.-Ing. Paul E.: Leitfaden der elektronischen Meßgrößenerfassung. Franzis-Verlag, München (nur noch über Verfasser erhältlich).

[4] Bürgel, Dipl.-Ing. Erdmann: Neue Normen und Schaltzeichen der digitalen Informationsverarbeitung. Franzis-Verlag, München.

[5] Code-Übersicht: ELEKTRONIK Arbeitsblatt Nr. 69, ELEKTRONIK, Heft 9, S. 327, 328; H. 10, S. 361 . . . 364, 1972.

[6] Codes für die Daten- und Nachrichtentechnik. ELEKTRONIK Arbeitsblatt Nr. 101, ELEKTRONIK, Heft 1, S. 81, 82; H. 2, S. 79, 80, 1977.

[7] DIN 66250, Entwurf Februar 1979: Zahlendarstellung für den Datenaustausch. Beuth-Verlag, Berlin und Köln.

[8] Normen für Größen und Einheiten in Naturwissenschaft und Technik. DIN-Taschenbuch 22, 1978. Beuth-Verlag, Berlin und Köln.

[9] Informationsverarbeitung 1, Normen über Begriffe, Sinnbilder und Schaltzeichen, maschinelle Zeichenerkennung, Papiere und Farbbänder für die Datenverarbeitung, Schnittstellen. DIN-Taschenbuch 25, 1978. Beuth-Verlag, Berlin und Köln.

[10] Informationsverarbeitung 2, Normen über Codierung, Datenträger, Programmierung. DIN-Taschenbuch 125, 1978. Beuth-Verlag, Berlin und Köln.

[11] Normen über graphische Symbole für die Elektrotechnik, Schaltzeichen und Bildzeichen. DIN-Taschenbuch 7, 1979. Beuth-Verlag, Berlin und Köln.

[12] Normen über die Schaltungsunterlagen für die Elektrotechnik. DIN-Taschenbuch 107, 1979. Beuth-Verlag, Berlin und Köln.

[13] Digital Research, CP/M Beschreibungen vertrieben z.B. durch Datamega.

[14] Dr. Dobb's Journal of COMPUTER Calistenics & Orthodontia
PCC
Box E
Menlo Park CA 94025.

[15] Z80 Assemblersprache, ZILOG
Benutzerhandbuch in deutscher Sprache
Vertrieb auch durch Kontron in München Eching.

[16] INTEL Datenhandbuch 1978
Intel Semiconductor GmbH
Seidlstraße 27
8000 München 2.

Bezugsquellenverzeichnis

1. Kristallverarbeitung
   Neckarbischofsheim GmbH
   6924 Neckarbischofsheim
   Quarze
2. Atlantik Elektronik GmbH
   Hofmannstraße 20
   8000 München 70
   CRT-Controler 5027
3. Motorola Semiconductors
   Münchner Straße 18
   8043 Unterföhring
   6845 CRT-Controler
4. Adcomp Datensysteme
   Horemannstraße 8
   8000 München 19
   Low-Cost Drucker
5. DATAMEGA KG
   Hermann-Oberth-Straße 7
   8011 Putzbrunn
   Drucker, Peripherie (Writehander, Digitalisierer)
   Software.

Fachausdrücke – Glossar

A

Access: Zugriff. Möglichkeit zum Beispiel zu einer bestimmten Speicherzelle zuzugreifen.

A/D Umsetzer: Analog-Digital Umsetzer. Eine analoge Größe wird in eine digitale umgewandelt.

Adresse: Eine Bezeichnung für einen bestimmten Speicherplatz oder eines Speicherbereiches.

Akkumulator: Es handelt sich um ein Register, über das arithmetische und meist auch logische Befehle ausgeführt werden können. Der Akkumulator hat direkte Verbindung mit dem Rechenwerk.

ALE: Adress Latch Enable. Übernahme einer Adresse in einen Speicher durch dieses Signal.

ALGOL: Algorithmic Language. Es handelt sich um eine Programmiersprache für technisch wissenschaftliche Probleme.
Sie ist auch auf Mikroprozessoren verfügbar.
(von Lifeboat für den 8080).

Alignment: Abgleich. Justierung.

ALU: Arithmetic Logic Unit. Rechenwerk. In diesem Teil des Rechners werden die arithmetischen und logischen Befehle ausgeführt.

APL: Eine Programmiersprache für den technisch-wissenschaftlichen Bereich speziell auch für den Dialogbetrieb.

APU: Arithmetic Processor Unit.
Ein Gleitkommarechner z.B.

ASCII: American Standard Code for Information Interchange. Ein häufig gebrauchter Code für Informationsübertragung. Auch mit ISO-7-Bit-Code bezeichnet (DIN 66003).

Assembler: Ein Übersetzungsprogramm, das eine maschinennahe Programmiersprache in den Maschinencode übersetzt.

Assoziativ-Speicher: Ein Speicher, bei dem der Zugriff nicht über eine bestimmte Adresse, sondern über den Speicherinhalt erfolgt.

Asynchron: Taktunabhängige Betriebsweise.

Available: Verfügbar.

B

Bankselekt: Kann der Speicher von einem Mikroprozessor nicht voll adressiert werden, so kann der Speicher in Bereiche (Banks) unterteilt werden, die dann durch einen eigenen Hardwaremechanismus ausgewählt werden.

BAS-Mischer: Aus den Synchronsignalen und dem Videosignal wird ein genormtes Signal zur Ansteuerung von einem Videomonitor erzeugt.

BASIC: Beginners All purpose Symbolic Instruction Code. Eine einfache dialogorientierte Programmiersprache, die in großer Vielfalt für Mikrocomputer erhältlich ist.

Baud: Messung des Datenflusses, wobei die Zeit zur Übertragung des kürzesten Elementes als Maß genommen wird. Führt jedes Element ein Bit, so ist die Baudrate zahlenmäßig der Anzahl Bits pro Sekunde.

Betriebsystem: Darunter versteht man eine Reihe von Programmen, die es dem Computer ermöglichen, selbständig Programme zu bearbeiten. CP/M ist zum Beispiel ein einfaches Betriebsystem für den 8080/Z80.

Bildwiederholspeicher: Bei einem Datensichtgerät ist dies der Teil in dem die darzustellenden Zeichen stehen. Er muß sowohl vom CRT-Controler als auch von einer weiteren Steuereinheit ansprechbar sein.

Bit: Binary Digit. Kleinste Informationseinheit.

Bootstrap-Loader: Urlader. Ein Programm, das es dem Computer ermöglicht, Programme zu laden. Dabei wird meist zunächst ein weiterer Lader geladen, der dann seinerseits in der Lage ist, größere Programme zu laden. So wird zum Beispiel das CP/M Betriebssystem geladen.

Branch: Verzweigung.

Buffer: Puffer. Speicher, in dem Daten kurzzeitig festgehalten werden oder auch Treiber zum Schalten von größeren Lasten.

Bubble memory: Magnetblasenspeicher.

Bus: Sammelleitung, an die mehrere Bausteine angeschlossen werden können.

Busy: Besetzt. Belegt. Beschäftigt.

Byte: 8 Bits.

C

Cartridge: Magnetbandkassette.

CCD memory: Ladungsgekoppelte Speicher.

Clock: Takt.

COBOL: Common Busines Oriented Language. Eine Programmiersprache für vorwiegend kaufmännische Aufgaben. Sie ist unter dem Betriebsystem CP/M auch für Mikrocomputer erhältlich.

Compiler: Ein Übersetzungsprogramm, das eine höhere Programmiersprache in den Maschinencode übersetzt.

Conditional: Bedingt.

Controler: Steuereinheit. Beim IEC-Bus ist es das Gerät, das auch Steuerinformation aussenden darf um z.B. Daten von anderen Geräten anzufordern.
Conversion: Übersetzung, Umkodierung.
Core: (Magnet-)Kern (Speicher).
CP/M: Disk Operating System für den 8080 von DIGITAL RESEARCH.
CPU: Central Processing Element.
Zentraleinheit eines Computers, bestehend aus Rechenwerk und Steuerwerk.
Cross-Assembler: Ein Assembler, der nicht auf der Maschine, für die er den Maschinencode erzeugt. lauffähig ist.
Es gibt so zum Beispiel einen Assembler für den 6800, der auf den 8080 unter CP/M läuft.
Cross-Compiler: Ein Compiler, der auf einer anderen Maschine läuft als auf der, für die er den Maschinencode erzeugt.
CRT: Cathode Ray Tube. Datensichtgerät oder Datensichtgeräte-Bausteine.
Cursor: Sichtmarke zur Kennzeichnung der aktuellen Schreibposition z.B. in Datensichtgeräten.

D

Daisy chain: Kette. Verkettung. Zum Beispiel um Interrupt-Prioritäten festzulegen, können von einem Baustein zum anderen Signale geführt werden, die also nicht nach dem Prinzip einer Busleitung arbeiten, sondern durch geographische Anordnung wird die Priorität festgelegt.
Data aquisition: Datenaufnahme. Datensammlung.
Data logger: Meist ein Gerät zur Aufnahme von analogen Daten und deren Umwandlung in eine digitale Darstellung. Die Daten können dann von einem Rechner abgeholt werden.
Debugging: Fehlersuche und Beseitigung. („Entwanzen").
Dense graphic: Mittlere Dichte bei Graphicdarstellungen. Z.B. 256 x 128 Punkte.
Density: Dichte. Zum Beispiel haben „double dense" Floppys doppelte Datendichte.
Device: Gerät, Einheit.
D/A-Umsetzer: Digital-Analog-Umsetzer.
Dialoggerät: Gerät zur direkten Datenein- und Ausgabe.
Digit: Ziffer, Stelle.
Digitalisierer: Ein Gerät zur Eingabe graphischer Darstellungen, z.B. mit einem Griffel oder Fadenkreuz auf einer speziellen Digitalisierungsoberfläche.
Direct-acess: Direkter Zugriff.
Direktory: Inhaltsverzeichnis z.B. von einer Floppy Disk.
DMA: Direct Memory Acess. Direkter Zugriff auf den Speicher eines Rechners, wobei die Zugriffssteuerung nicht vom Rechner vorgenommen wird, sondern zum Beispiel von einer Peripherie-Einheit.
DOS: Disc Operating System. Ein Programm, das es ermöglicht, mit einer Floppy oder einem Plattenspeicher zu arbeiten. CP/M(8080/Z80), FLEX(6800/6809) oder ISIS(8080) sind solche DOS.
Dreileiterhandshake: Wird z.B. beim IEC-Bus verwendet um einen Datenaustausch zu synchronisieren.
Drum storage: Trommelspeicher.
Dump: Auszug eines Speicherinhalts.
Durchsatz: Anzahl der Operationen, die ein Computer in einer Zeiteinheit leistet.
Dynsmische Speicher: Bei solch einem Speicher muß die Information zyklisch aufgefrischt werden. Vorteil ist die Verfügbarkeit von hohen Speicherkapazitäten.

E

EAROM: Electrical Alterable Read Only Memory. Festwertspeicher, der sich nicht nur elektrisch programmieren, sondern auch löschen läßt.
EBCDIC-Code: Extended Binary Coded Decimal Interchange Code. Ein Alphanumerischer 8-Bit-Code.
Editor: Ein Programm zur Eingabe von Texten, Änderung und Ausgabe für Source Programme oder Textverarbeitung.
ELIZA: Ein Dialogprogramm, das einen Psychologen darstellt und zur Erprobung von künstlicher Intelligenz erstellt wurde.
Emulation: Sofwaremäßige Nachbildung eines Computers, so daß der Befehlssatz eines Computers, der nachgebildet wird; auf einem anderen verfügbar ist, wenn auch die Ausführungszeit im allgemeinen kleiner ist als auf dem realen Computer.
Enable: Freigabe.
to enter: Eingeben.
EOC: End of Conversion. Ein Signal, das das Ende einer Umwandlung anzeigt.
EPROM: Erasable Programmable Read Only Memory. Ein mit ultraviolettem Licht löschbarer, aber elektrisch programmierbarer Festwertspeicher.
to erase: Löschen.
Error: Fehler. Irrtum.
Europakarte: Leiterplatte mit genormtem Format: 100 mm x 168 mm.
Evaluation module: Entwicklungseinheit.
Even-odd parity: Gerade oder ungerade Parität (Quersumme binär).
Exorciser: Hilfsgerät zur Entwicklung von Mikrocomputersystemen.
Expression: Ausdruck.

F

Fan-in: Eingangslastfaktor.
Fan-out: Ausgangslastfaktor. Er gibt an, wie viele Bausteine der gleichen Logikserie an einen Ausgang mit dem angegebenen Fan-out angeschlossen werden können.
Fifo: First In First Out. Zuerst eingehende Daten werden auch zuerst wieder ausgegeben.

File: Datei. Daten. Eine Ansammlung von Datengruppen, die in einer Datei abgelegt werden.
Firmware: Eine Software, die fest zur Funktionsfähigkeit eines Systems z.B. in einem ROM vorhanden ist.
Fixed-point: Festkomma.
Flag: Eine Marke oder ein Flip-Flop zum Festhalten eines Zustands. Merker.
Floating-point: Gleitkomma.
Floppy-Disk: Ein billiger Massenspeicher mit Kapazitäten von 90K Bytes (Minifloppy) bis 2M Byte (Große Floppy mit Quad dense etc.).
Fortran: Formula Translation. Eine problemorientierte Programmiersprache, die für technisch-wissenschaftliche Aufgaben ausgelegt ist. Für Mikrocomputer gibt es eine Vielzahl von Fortran Compiler.
Frontpanel: Bedienungsfeld.

G
Gate: Verknüpfungsschaltung.

H
Handler: Routine zur Kontrolle eines peripheren Gerätes.
Handshake: Quittungsbetrieb. Methode, um Geräte mit verschiedenen Arbeitsgeschwindigkeiten durch den Austausch von Steuersignalen zu synchronisieren.
Hardcopy: Kopie. Zum Beispiel ein direkter Ausdruck des aktuellen Bildschirminhalts.
Hardware: Damit sind alle Bauteile, Geräte eines Systems gemeint.
Hexadezimal: Siehe sedezimal.
High order: Höherwertige Stelle.

I
ICE: In-Circuit Emulator. Mit Hilfe einer Nabelschnur, an dessen Ende z.B. ein Sockel für die Z80 CPU ist, kann von einem Entwicklungssystem aus der Ablauf in einem Mikroprozessoranwendungssystem verfolgt werden.
IEC-Bus: Schnittstellennorm, um mit 8 Bit parallel und Byteseriell Daten austauschen zu können.
Include: Beinhalten. Einschließen.
Increment: Schrittweises erhöhen eines Wertes.
Index-Register: Ein Register zur Modifikation der Operandenadresse eines Befehls. Damit ergibt sich z.B. durch Addition des Inhalts des Index-Registers zum Adreßteil eines Befehls eine neue effektive Adresse.
Initialisierung: Die Anfangsschritte in einem Programm, um definierte Startwerte zu erhalten.
Input: Eingabe.
Instruction: Befehl, Anweisung.
Interface: Schnittstelle. Mit Hilfe eines Interface können zwei Systeme einander angepaßt werden.
Interpreter: Ein Interpreter ist ein Programm, das z.B. Befehle einer höheren Programmiersprache direkt ausführt und nicht zuerst vollständig in Maschinencode übersetzt.
Interrupt: Unterbrechung. Durch einen Interrupt, den meist ein Peripheriegerät anfordert, wird das gerade laufende Programm unterbrochen und eine spezielle Unterbrechungsroutine ausgeführt. Danach erfolgt ein Rücksprung in das unterbrochene Programm.

J
Job: Auftrag.
Joystick: Ein Kreuzknüppelpotentiometer, das zur Eingabe von Daten verwendet werden kann. Vorwiegend in Kombination mit graphischen Geräten.
Jump: Sprung.
Jumper: Leitungsbrücke. Mit Jumper wird im Jargon eine Brücke zur Einstellung von Parametern, wie z.B. einer Perpherieadresse verwendet.

K
Keyboard: Tastatur.
Kit: Bausatz.
Kompatibel: Austauschbar, aneinander angepaßt.

L
Label: Marke. In Programmiersprache ist damit meist eine symbolische Adresse gemeint. Auf Magnetbändern z.B. ist damit ein Identitäskennzeichen gemeint.
Lichtgriffel: Ein Stift, der zur Eingabe von Daten direkt über die Bildfläche geeignet ist. Dazu liefert er immer dann einen Puls, wenn der Schreibstrahl des Sichtgerätes auf die Optil des Griffels trifft.
Lifo: Last In First Out. Zuletzt gespeicherte Daten werden zuerst ausgegeben. Stack.
Linker: Ein Programm, das mehrere Teilprogramme, die schon assembliert wurden, zu einem binden kann. Dazu muß der Assembler die nötige Information dem Linker übergeben können.
LISP: List Processing. Eine Programmiersprache für die Verarbeitung von Listen und rekursiven Datenstrukturen. Die Sprache eignet sich besonders für AI (artificial intelligenc – künstliche Intelligenz). Das bekannte Programm ELIZA wurde z.B. zunächst in LISP programmiert.
Listener: Name der beim IEC-Bus verwendet wird. Ein Gerät, das Daten vom IEC-Bus empfangen kann, heißt Listener.
Listing: Ausdruck. Auflistung.
Loader: Ein Ladeprogramm.
Logic analyzer: Ein Hilfsgerät zum Testen von Digitalschaltungen mit einer Anzeige für die logischen Zustände in dieser Schaltung.
Loop: Schleife. Durch einen Sprung zurück kann zum Beispiel eine Schleife entstehen.
Low order: Niederwertige Stelle.

M

Makro: Mit Hilfe einer Makroanweisung kann der Programmierer in Assemblern eine Folge von Befehlen definieren, die durch Angabe des Makronamens in dem Assemblerprogramm durch den Assenbler bei der Übersetzung eingefügt werden. Dabei können durch Parameterangaben die eingefügten Befehlssequenzen variiert werden.

Maschinencode: Maschinensprache. Damit ist ein Binär-Code gemeint, der vom Mikrocomputer z.B. direkt verstanden wird.

Maske: Ein Bitmuster, mit dem bestimmte Bitgruppen ausgeblendet, komplementiert oder eingefügt werden.

Memory: Speicher.

Mikroprogrammierbar: Der Befehlssatz eines Prozessors kann mit Hilfe von Mikrobefehlen definiert werden.

Mikrocomputer: Besteht aus einem Mikroprozessor, Speichern und Peripherie.

Mikroprozessor: Ein integrierter Baustein, als Teil eines Mikrocomputers, der ein Leit- und ein Rechenwerk besitzt. Der interne Ablauf kann in der Regel von außen durch Software beeinflußt werden.

Mnemonischer Code: Leicht zu merkende Kurzwörter, deren Inhalt auf die Verwendung schließen läßt. Derartige Kurzwörter werden in Assemblersprachen eingesetzt.

Modem: Modulator und Demodulator. Eine Schaltung, die Daten für die Fernübertragung aufbereitet.

Monitor: Ein Programmsteuersystem, das auch aus Hardware bestehen kann.

Multiplex: Übertragung von mehreren verschiedenen Informationen, die dazu zeitlich hintereinander übertragen werden.

Multiprozessing: Ein aus mehreren CPUs oder Teilcomputern zusammengesetzter Rechner.

N

Nesting: Verschachtelung. Zum Beispiel verschachteln von Interrupts.

O

Off-line: Der Benutzer ist dabei nicht hardwaremäßig mit dem Computer verbunden, sondern der Verkehr wird über Datenträger abgewickelt.

Oktal: Zahlendarstellung zur Basis 8.

On-line: Dabei ist das Terminal des Benutzers über eine Datenleitung direkt mit dem Computer verbunden.

Output: Ausgabe.

P

Packen: Dabei werden zum Beispiel zwei Dezimalzahlen in einem Byte untergebracht.

Parity: Parität. Gleichheit.

PASCAL: Eine höhere Programmiersprache, die zunehmend Verbreitung auch bei Mikrocomputern findet.

Pass: Lauf. Zum Beispiel eines Programms.

Peripherie: Externe Datenend- und Speichergeräte.

PIA: Peripheral Interface Adapter. Ein Baustein, der den Ein- und Ausgabeverkehr zwischen dem Mikroprozessor und der Peripherie abwickelt.

Pipelining: Fließbandverarbeitung. Durch diese Verarbeitungsform kann die Ausführungszeit stark verkürzt werden. Während ein Befehl gerade ausgeführt wird, wird der nächste Befehl schon geholt. Bei Sprungbefehlen ergeben sich im allgemeinen zusätzliche Verzögerungen.

PL/1: Programming Language. Eine höhere Programmiersprache, die ebenfalls für Mikrocomputer erhältlich ist.

PL/M: Programming Language for Microcomputers. Eine höhere Programmiersprache, die auf der Basis von PL/1 arbeitet und speziell für Mikrocomputer entwickelt wurde.

Pointer: Zeiger. Ein Speicherplatz, der eine Adresse enthält. Mit einem Zeiger lassen sich leicht Stacks aufbauen. (Stack-Pointer).

Polling: Aufrufbetrieb. Aufruftechnik. Um z.B. die Quelle eines Interrupts festzustellen, werden alle in Frage kommenden Quellen abgefragt. Dieser Vorgang wird mit polling bezeichnet.

Port: Tor. Schaltkreise wie Register z.B. für die Ein-/Ausgabe.

Power on jump: Nach Einschalten der Stromversorgung wird ein Sprungbefehl durch eine Hardwareschaltung vorgenommen, um z.B. in ein Monitorprogramm zu gelangen, dessen Anfangsadresse nicht mit der Adresse übereinstimmt, die nach einem Reset vom Prozessor angewählt wird.

Programm: Ist eine Folge von Anweisungen (Befehlen), die zur Lösung eines Problems dienen sollen.

Programmierbarer Zeichengenerator: Der Zeichensatz kann durch den Prozessor frei programmiert werden. Damit ist es einfach möglich, Sonderzeichen, Schaltsymbole im Programm selbst festzulegen und zur Darstellung zu gebrauchen.

Programmiersprache: Eine Sprache zur Formulierung von Programmen, die automatisch in die Maschinensprache übersetzt werden können.

PROM: Programmable Read Only Memory. Ein programmierbarer Festwertspeicher.

Pseudobefehl: Eine Instruktion, die eigentlich gar nicht vorhanden sein dürfte, gemäß den Herstellerangaben.

to punch: Stanzen, Lochen.

Q

Queue: Warteschlange. Daten werden in einer Warteschlange angesammelt, wenn sie noch nicht verarbeitet sind.

R

RALU: Register and Arithmetic Logic Unit. Ein Prozessorelement mit einer ALU und einigen Registern.
RAM: Random Access Memory. Ein Schreib-/Lesespeicher mit wahlfreiem Zugriff.
Reader: (Lochstreifen- oder Lochkarten) Leser.
Real-Time: Echtzeit. Arbeitsweise eines Computers.
Real time clock: Echtzeituhr.
Redundanz: Teil einer Nachricht, die zur eigentlichen Information nichts mehr beiträgt. Sie kann aber zum Beispiel zur Fehlererkennung oder Korrektur verwendet werden.
Refresh: Wiederauffrischung. Wird bei dynamischen Speichern benötigt, um einen Informationsverlust zu verhindern.
Relokalisierbar: Ein Programm, das auf verschiedenen Speicherplätzen direkt lauffähig ist heißt relokalisierbar.
REPROM: Reprogrammable Read Only Memory. Ein Festwertspeicher, der sich löschen und wieder neu programmieren läßt.
Request: Anfordern. Anforderung.
to reset: Rücksetzen, in den Grundzustand bringen.
Resident: Im eigenen System läauffähig. Zum Beispiel bei einem resident assembler.
ROM: Read Only Memory. Ein Festwertspeicher von dem nur gelesen werden kann.
Run: Durchlauf.

S

to scan: Abtasten.
Schnittstelle: Pegel- und Anschlußgenormte Trennstelle zwischen zwei Geräten.
sedezimal: Zahlendarstellung zur Basis 16.
(0 1 2 3 4 5 6 7 8 9 A B C D E F).
Häufig auch mit Hexadezimalsystem bezeichnet.
to select: Auswählen.
to sense: Abtasten.
Simulator: Ein Programm, das einen Befehlssatz simuliert.
Software: Hierunter versteht man alle Arten von Programmen, wie auch Texte und Information.
Source: Quelle.
Space: Freiraum.
Sprase graphic: Eine niedrig auflösende Graphic. Z.B. 128 x 128 Punkte.
Stack: Stapelspeicher, Kellerspeicher. Merkmal für einen Stack ist, daß eine Informationseinheit immer nur an der Stelle entnommen werden kann, an der sie gerade hinzugefügt wurde. (LIFO).
State: Zustand. Operationsschritt.
Statement: Anweisung. Befehl.
Statischer Speicher: Ein Speicherbaustein, der keine Wiederauffrischzyklen benötigt.
Steuerwerk: Dieser Teil im Computer kontrolliert die Ausführung sämtlicher Befehle. Er wird auch mit Leitwerk bezeichnet.
Subroutine: Unterprogramm.
Super dense graphic: Hoch auflösende Graphic. Z.B. 256 x 256 Bildpunkte.
Supervisor: Ein Organisationsprogramm.
Synchron: Ein Takt steuert den genauen Ablauf.
Syntaxdiagramme: Eine besondere Art in Diagrammen, Sprachbeschreibungen durchzuführen. Wurde durch die PASCAL Syntaxdiagramme bekannt.

T

Talker: Name aus der IEC-Bus-Definition. Ein Gerät, das Daten auf diesen Bus senden kann, heißt Talker.
Tape: Ein Magnetband oder Lochstreifen.
Terminal: Datenendstation. Ein Gerät zur Daten-ein- und/oder Datenausgabe.
Text-Editor: Siehe Editor.
Time sharing: Zeitscheibenverfahren. Dabei können mehrere Benutzer auf ein und denselben Computer zugreifen.
Timing: Zeitablauf.
Tiny: Klein. Tiny BASIC zum Beispiel bedeutet eine Teilmenge des Standard BASICs, ein BASIC mit eingeschränktem Befehlssatz.
Trace: Ablaufverfolgung. Ein Programm kann durch die schrittweise Ausführung und Protokollierung überwacht und damit ein Fehler leichter gefunden werden.
Track: Spur. Bahn. Eine Floppy ist beim 8-Inch-Format in 76 Tracks unterteilt.
to transfer: Übertragen.

U

UART: Universal Asynchronous Receiver/Transmitter. Diese Schaltung übernimmt die Serien/Parallel- sowie die Parallel/Serienwandlung für eine asynchrone Datenübertragung.
Unit: Gerät. Einheit.
Unterprogramm: Gleiche Befehlsfolgen, die in einem Programm mehrmals vorkommen werden, im allgemeinen als Unterprogramm ausgeführt, so daß diese nur einmal im Programm auftreten und mit einem Unterprogrammsprung vom Hauptprogramm aufgerufen werden.

V

V24: Schnittstellen-Norm für serielle Signale.
Valid: Gültig.
Vektor Interrupt: Von dem Gerät, das den Interrupt an den Prozessor gibt, wird zusätzlich ein Interruptvektor mitgeliefert, der dem Prozessor sagt, welche Unterbrechungsroutine ausgeführt werden soll.
Volatile: Flüchtig.

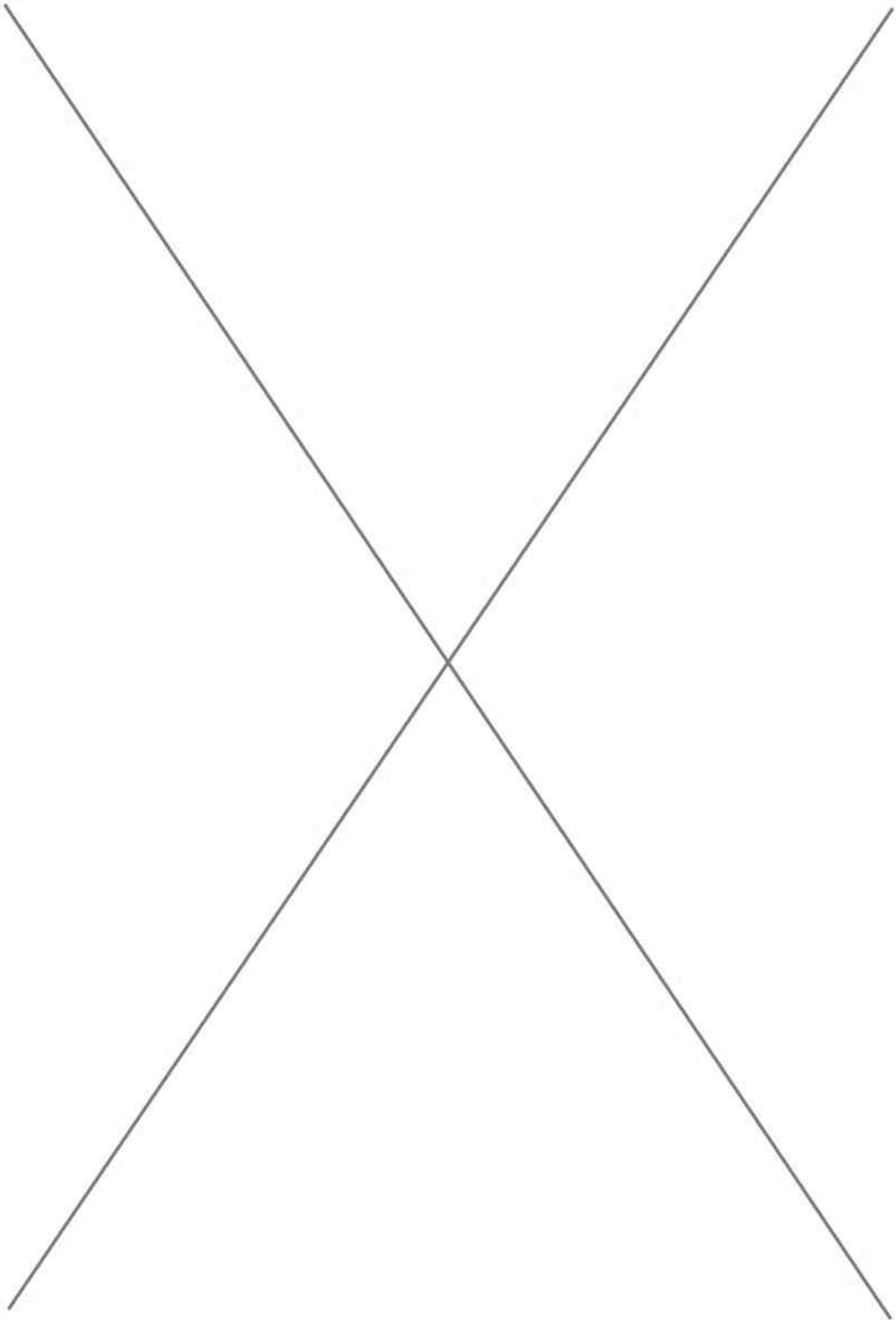
W

Winchester Drive: Ein spezielles Verfahren, bei dem es durch hermetischen Abschluß gelingt, hohe Speicherdichte auf kleinem Platz zu erreichen. Z.B. 8-inch-

Platte von IMI mit 22 M Bytes Kapazität auf drei Oberflächen.
Worts case: Ungünstigster Fall.
Wort: Zusammenfassung mehrerer Bits; sie können meist auch zusammenfassend verarbeitet werden. (Z8000 CPU z.B. hat eine Wortbreite von 16 Bit).

X
XY-recorder: Ein XY-Schreiber.

Z
Zeichengenerator: Meist ein Festwertspeicher, in dem ein Zeichensatz binär, z.B. in einer Matrixform gespeichert ist.
Zugriff: Zugang z.B. zu einer bestimmten Speicherzelle.
Zyklus: Eine Anzahl von Schritten, die wiederholt werden und im Ablauf gewisse Ähnlichkeiten aufweisen.



Sachverzeichnis

A
A/D Umsetzer 89
A/D Umsetzer Programm 91, 92
Anpassung des BASIC Interpreters 157
Assembler 139 ff.
– Mnemonics 31

B
Bankselekt 41
BAS-Mischer 73
BASIC 144 ff.
– Befehlssatz eines 12 K BASICs 161
– Befehlssatz des RDK BASICs 145
Baugruppen für CRT 71
Befehlsholphase 13
Befehle des Editors 122
Befehlssatz des RDK Monitors 102
– eines kommerziellen Monitors 117
Befehlstabelle Z80 25 ff.
Bildwiederholspeicher 72, 74
Blockschaltbild eines Datensichtgerätes 72
Boolscher Formelanalysator 200 ff.
Busanforderung 16
Bustreiber Anschluß 20

C
CP/M Betriebssystem 179
CRT 5027 50 ff.
– 6845 54 ff.
– - Controler 49
– - Display Programm 59
Cursormöglichkeiten 57

D
D/A Umsetzer 96
Datensichtgeräte 49 ff.
Dense Graphic 68
Digitalisierer 99
Direktory 180
Disassembler 164 ff.
Displays 49 ff.
Dreileiterhandshake 78
Drucker 76 ff.
– programm für Minidot Druckwerk 82
– programm für Parallel-drucker 78
– programm für Serienschnittstelle 80
– mit Serienschnittstelle 79

E
EATER 197
Editor 122
Ein/Ausgabezugriffe 15
Einzelschritt für Z80 22
Erweiterungen des Editors 137

F
Flagregister 10
Floppy Disks 179
Funktionsweise des Editors 124
– des Monitorprogramms 103

G
GLIDER 197
Graphikschaltungen 62 ff.

H
Halt Zustand 18
HF-Generator 75

I
Interruptverarbeitung 16

J
Joystick Anschluß 93

K
Kassettengerät als Datenspeicher 84

L
Lader für TDL
Relocating Format 179
LIFE 192
Linking Assembler 141

M
Makro Assembler 142
Mikrofon Anschluß 93
Minidot Druckwerk 81
Minimalsystem 23
Monitorprogramme 102 ff.

P
Parallel|drucker 76
– ports 44
Phase encoding Technik 86
Power on jump 42
Programmierbarer Zeichengenerator 70
Pseudobefehle 26

Q
Quarzgeneraotr 19

R
RAM Zugriff 14
Regenerierverstärker 96
Registerstruktur 9
Relocator 182
Relocating Assembler 140
Reset Sprung 42
– - Logik 19

S
Schnittstellen 48
Serienports 46
Softwaretracer 172
Sparse Graphic 67
Speicher|bankselekt 40
– erweiterung RAM 37
– erweiterung ROM 39
Sprachaufzeichnung 94
Spracherkennung 94
Sprechererkennung 94
Super Dense Graphic 69
Syntaxdiagramme 209

T
Taktgenerator 19
TDL-Monitor 118
Testhilfen für Software 170
Timing 12 ff.
Tracer 172

U
Universal Disassembler in BASIC 164

V
V24/28 mA Treiber 48

W
Wait-Zyklus-Erzeugung 21 ff.
Writehander 97

Z
Z80 Befehlstabelle 25 ff.
– Bustreiber Anschluß 20
– Hardware 9
– Pin Belegung 10
– Pin Beschreibung 11
– Pseudobefehle 26
– Timing 12 ff.
Zeichengenerator 63

Klein
Mikrocomputer
Hard- und Softwarepraxis

Hier geht es um den Z80, das Arbeitspferd unter den Mikrocomputern. Ihn hardwaremäßig gut zu verstehen, ihn softwaremäßig voll auszufahren, das bewirkt dieses Buch.
Der Band besteht aus drei Teilen, die deutlich miteinander verknüpft sind.
Erstens, dem Hardware-Teil. Darin bespricht der Autor die zahlreichen Peripheriegeräte, die einen Z80 erst zum vollwertigen Mikrocomputer machen. Als da sind: Serienwandler. CRT-Controler. Druckeransteuerung. A/D- und D/A-Umsetzer.
Zweitens, dem Software-Teil. Hier wird die wichtige System-Software so intensiv besprochen, daß sie dem Anwender möglichst schnell geläufig wird. Monitorprogramme. Editor. BASIC mit Abwandlungen und Vereinfachungen. Assembler. Softwaretracer in Maschinensprache. Ausgedehnte Befehlslisten, praxisorientiert, sind für diesen Buchteil charakteristisch.
Drittens, dem Programmteil. Der lehrt das freie Programmieren, wiederum mit ausführlichen Programmlisten. Diese vermitteln dem Benutzer einen großen Erfahrungsschatz, den andere eingesammelt haben.

Rolf-Dieter Klein

ISBN 3-7723-6811-5

Franzis'